Títol: Daurum v.5.0: una aplicació per a Record Linkage

Volum: 1
Alumne: Robert Ventura Simon

Director: Josep Lluís Larriba Pey
Departament: Arquitectura de Computadors (AC)
Data: 29 de Gener de 2010

DADES DEL PROJECTE

Títol del Projecte: DAURUM v.5.0: Una aplicació web per a Record Linkage

Nom de l'estudiant: Robert Ventura Simon

Titulació: Enginyeria en Informàtica

Crèdits: 37.5

Director: Josep Lluís Larriba Pey

Departament: Arquitectura de Computadors (AC)

MEMBRES DEL TRIBUNAL *(nom i signatura)*

President: Victor Muntés Mulero

Vocal: Josep Ramón Herrero Zaragoza

Secretari: Josep Lluís Larriba Pey

QUALIFICACIÓ

Qualificació numèrica:
Qualificació descriptiva:

Data:

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Projecte de Final de Carrera

DAURUM v.5.0: Una aplicació web per a Record Linkage

Alumne:
Robert Ventura Simon

Director:
Josep Lluís Larriba Pey

Agraïments

Primerament, m'agradaria expressar el meu agraïment a totes aquelles persones, que d'una manera o altra, m'han ajudat en la realització d'aquest projecte.

Al Josep Lluís Larriba Pey, el Director del projecte, per haver-me donat la possibilitat d'incorporar-me a DAMA-UPC i dur a terme aquest projecte ajudant-me quan ha fet falta.

A la Raquel Pau Fernández, companya de DAMA-UPC, per la constant ajuda que m'ha ofert i tots els bons consells que m'ha donat durant el projecte.

Al Francesc Escalé Claveras, company de DAMA-UPC, que m'ha ajudat a entendre millor el funcionament de Daurum i del Record Linkage.

A tots els companys de DAMA-UPC, l'Aleix, el Joan, el Miquel, el Norbert, la Núria, el Pere B., el Pere U., el Sergio, la Vanesa i els que ja no hi són, per haver pogut gaudir amb ells de bons moments durant la realització d'aquest projecte en un ambient de treball molt agradable i per l'ajuda que també m'han proporcionat.

Als meus amics, per haver fet més portable aquest temps de treball i pels ànims que m'han donat.

A la Manoli i el Jaume, els meus pares, ja que sense el seu suport tant moral com econòmic no hagués estat possible realitzar i acabar aquesta carrera. A tota la meva família també vull agrair l'amor que m'han brindat; en especial al meu germà Xavi, pels coneixements que m'ha aportat i les llargues estones de xerrades que hem tingut de camí a la feina, i a les meves àvies, Fina i Rosa, pel recolzament que m'han donat malgrat no entendre la feina que faig.

I finalment a la Doris, la meva xicota, per estimar-me tant, animar-me en tot moment amb el seu positivisme i per ajudar-me en tot el que he necessitat durant el projecte tot i no ser del seu àmbit.

Índex

Capítol 1

Idea de projecte

Aquest primer capítol introdueix alguns conceptes necessaris per a la comprensió del projecte i exposa els objectius que cal haver complert un cop s'hagi finalitzat. Principalment doncs, es dóna una breu definició que ajudarà a situar-nos en la temàtica del projecte i el seu futur desenvolupament.

1.1 Introducció

En la actualitat, les eines informàtiques són utilizades per una gran part de la societat i permeten cobrir necessitats que, en temps anteriors, eren molt costoses i complicades d'assolir. Les bases de dades han permés a les organitzacions emmagatzemar informació útil sobre els seus clients, productes i proveedors de manera ràpida i eficaç; però a vegades és necessari comprovar la seva consistència.

És fàcil imaginar-se una entitat pública com podria ser un Departament de Salut, la qual temps enllà va decidir fer ús dels progressos de la tecnologia per a introduir tots els seus membres dins d'una base de dades. Poc a poc, el volum de dades ha anat fent-se més i més gran i, arribat el moment de fer un anàlisi d'aquestes, es descobreix que hi ha diferents entrades que corresponen a una mateixa persona i, per tant repetida, però amb petites diferències a l'hora d'escriure els noms.

Degut a un mal ús de la tecnologia, la base de dades no ha estat capaç de detectar aquestes repeticions. Si s'hagués dissenyat la base de dades amb una clau primària, la qual ja existia degut a que les persones membres del Departament de Salut posseeixen un identificador únic (CIP), aquest error s'hagués solucionat fàcilment.

Ens trobem doncs, davant d'un repte que consisteix en trobar aquelles entrades d'una base de dades que corresponen a una mateixa entitat. Aquest procés és intractable si s'intenta fer de forma manual, degut als grans volumns de dades que normalment representen. És aquí on apareixen les tècniques que s'anomenen **Record Linkage** (en endavant RL). Aquestes tècniques tenen com a finalitat emparellar aquelles entrades que tenen un cert grau de semblança. En finalitzar el procés s'obté un conjunt de parells d'entrades (similituds) que cal que un expert decideixi si, realment, corresponen a una mateixa entitat.

Un altre cas en el que és important identificar entrades similars o idèntiques és en la fusió de dades. Les grans empreses solen subministrar els seus productes als centres destinataris mitjançant distribuidors, els quals tenen assignats una zona de repartiment. L'empresa rep per part de cadascun dels distribuidors un registre amb les dades dels centres on s'ha fet alguna entrega; però és força habitual que aquests distribuidors incompleixin amb la seva assignació i vagin a entregar a zones que no els hi pertoca. Per a no ser descoberts, varien lleugerament les dades del destí, com per exemple l'adreça, el telèfon o fins i tot el nom on s'ha fet una entrega.

Posteriorment, l'empresa introdueix tots els registres rebuts per part dels distribuidors a una base de dades, la qual tampoc detecta la mala pràctica utilitzada per alguns d'ells i, per tant, no s'assabenta de la picaresca utilitzada.

Un altre cop, un bon ús de la tecnologia utilitzant identificador únics hagués solventat el problema i s'ha de recórrer al RL per a fer-ho.

El Record Linkage resulta molt útil doncs per a netejar o fusionar bases de dades, ja que detecta possibles similituds, però aquestes no son les úniques utilitats que té. Un altre àmbit en el qual pot ser útil aplicar-lo és en el creuament dades. Per exemple, els hospitals sovint fan seguiment dels seus pacients segons la malaltia que tenen per a obtenir estadístiques. El problema és que els malalts no sempre romanen en el centre i és habitual que acabin morint a casa seva si tenen una malaltia terminal. Així doncs, és difícil que el centre hospitalari s'assabenti de la seva mort i, per tant, no és possible actualitzar el seguiment que s'ha estat fent del pacient. L'única forma d'assabentar-se és consultant el Registre de Mortalitat: si es creuen les dades de mortalitat amb les dades del seguiment de l'hospital, és possible descobrir quins dels pacients, als que s'ha estat fent seguiment, ja han mort. Per a fer aquest creuament es poden també utilitzar tècniques de RL.

Finalment, un altre àmbit en el que pot ser útil aquesta tècnica és en la necessitat actual de fer anònimes les bases de dades, que significa ser capaços de mantenir les propietats estadístiques d'una base de dades sense que ningú pugui relacionar cadascuna de les entrades d'ella amb l'entitat real a la que correspon. El Record Linkage es pot emprar precisament per a avaluar si el procés d'anomització d'una base de dades és eficaç o no comparant les dades resultants amb les reals [18].

El funcionament del procés de Record Linkage és el següent: a partir d'un fitxer de registres com el que es pot veure a la Figura 1.1 es comparen els valors dels seus atributs (columnes) com ara Nom, Cognoms, Id, Adreça, ... per a trobar el grau de similitud que hi entre cadascun d'ells.

Nom	Cognom 1	Cognom 2	Edat	Sexe	Població
Jesús	Santana	Masdeu	33	H	Arbúcies
Josep M.	Subirats	Puigfalcó	65	H	Barcelona
Josep Maria	Sovirats	Puigfalcó	65	H	Barcelona
Ramon	Solé	Recasens	83	D	Vic
Ramona	Solé	Recasens	93	D	Vic
Jesep M.	Subirats	Puigfalcó	65	H	Barcelona
Roger	Subirats	Garcia	21	H	Girona
Rosalia	Subirats	Gracia	23	D	Mataró

Figura 1.1 : Exemple de fitxer amb sis atributs: Nom, Cognom1, Cognom2, Edat, Sexe i Població i vuit registres.

Així doncs, es pot veure que cadascún dels registres s'ha de comparar amb la resta per a poder trobar un grau de similitud entre ells. Per tant, el cost d'aquest procés és quadràtic $O(N^2)$,on N és el nombre de registres que conté la base de dades amb la qual s'està treballant. Cal tenir en consideració també, que és usual que el volum de dades sigui gran i que el temps esdevingui una restricció important ja que aturar el sistema de gestió de la informació pot implicar deixar d'oferir els serveis que la organització en qüestió dóna als seus usuaris. Tenint en compte aquesta premisa, és necessari, doncs, poder accelerar el procés de Record Linkage. En la literatura es poden trobar mètodes que permeten reduir la complexitat del problema.

1.2 Motivació

Actualment, el grup DAMA-UPC [7] proporciona al públic un software de Record Linkage anomenat Daurum. Es tracta d'una aplicació d'escriptori, en la qual es poden introduir fonts de dades i, després d'un procés de configuració, es fa l'execució del mètode de RL i s'obté una graella amb entrades aparellades i un cert grau de similitud representat per un percentatge. A partir d'aquest punt, l'usuari decideix quines parelles són acceptades i quines no. A més, una vegada acabada aquesta tasca, es permeten generar fitxers d'exportació en diversos formats per a una posterior utilització.

El fet de ésser una aplicació escriptori i que, per tant, cada usuari del sistema l'hagi de tenir instal·lada a la seva màquina suposa que, malauradament, la tasca de revisió de les similituds trobades no pugui ser dividida i realitzada per diversos usuaris a la vegada. A més, el manteniment i generació de fitxers no centralitzats pot provocar una pèrdua d'informació o inconsistència entre dades.

Aquest PFC neix doncs de la intenció de desenvolupar una nova versió per tal d'oferir una solució a les mancances trobades en l'aplicació descrita amb anterioritat.

1.3 Definició

L'objectiu principal d'aquest projecte és la realització d'una nova eina informàtica que permeti executar realitzar un procés de Record Linkage a partir d'unes dades entrades, es puguin després revisar les similituds trobades i finalment exportar-les.

Això vol dir doncs, que l'aplicació haurà de permetre analitzar les dades desitjades per a trobar els registres que corresponguin a una mateixa entitat física (similituds) segons els criteris escollits pels usuaris. A més, haurà de disposar d'una visualització clara, ordenada i intuïtiva dels resultats per a que aquests puguin ser avaluats de forma fàcil i exitosa i també haurà de permetre exportar a fitxers les dades que més convinguin sobre aquests resultats.

Per últim, haurà de resoldre les mancances detectades a l'anterior versió de Daurum, en la qual no està permesa la interacció entre usuaris i en la que llavors pot donar-se el fet de que es tinguin dades inconsistents. Així, aquesta aplicació permetrà que les tasques es facin de forma més col·laborativa, centralitzant-ne la seva informació per a un millor funcionament i una major eficàcia. També s'inclourà una gestió d'usuaris, la qual serà útil per a controlar l'accés a l'aplicació i les modificacions en les dades d'aquesta. Al Capítol 3.3 es pot veure més a fons quins tipus d'usuaris i quins privilegis tindrà cadascun d'ells.

1.4 Situació inicial

El grup de DAMA-UPC, format per desenvolupadors i investigadors, proporciona actualment al públic la versió 4.2 de Daurum. Aquest sofware utilitza una llibreria que conté dos algorismes de Record Linkage (Standard Blocking i Sliding Window, *vegeu Capítol 2.3*). La situació inicial del projecte serà doncs l'existència de la llibreria esmentada, de la qual es podrà fer ús degut a que no es té intenció de fer canvis en els mètodes que aquest proporciona. Pel que fa a l'aplicació en sí, no se n'aprofitarà ni les visualitzacions ni el model.

Així doncs, el projecte es centra en el desenvolupament d'una aplicació totalment nova que farà ús d'aquesta llibreria d'algorismes de RL. Això implicarà dur a terme un procés d'especificació i disseny complerts per tal de definir detalladament la construcció d'una eina que donarà solucions a les deficiències trobades en versions anteriors.

Capítol 2

Record Linkage

Aquest segon capítol servirà per tenir una visió més àmplia del Record Linkage, entendre quins són els reptes que representa a nivell de rendiment de les aplicacions i conèixer algunes de les propostes per a augmenta-ne el rendiment que es poden trobar a la literatura.

2.1 Contextualització

A finals de la dècada dels quaranta es va exposar per primera vegada la idea de Record Linkage, pocs anys després se'n van publicar els fonaments probabilístics de la teoria moderna i més tard se'n va formalitzar un model matemàtic [17] que continua sent vàlid i utilitzat per moltes aplicacions de RL existents avui en dia.

El procés de Record Linkage s'engloba dins del conjunt de tècniques de re-identificació, les quals s'apliquen dins del camp de la gestió de la informació i estan centrades en establir relacions entre les diferents entitats que hi ha en les diferents fonts de dades. L'obtenció de relacions entre entitats pot tenir sentit en escenaris com:

- Schema matching [20]: Aquest és un problema comú en moltes aplicacions basades en dades i consisteix en, donats dos esquemes diferents, trobar les relacions que hi pot haver entre els seus atributs.
- Data integration [15]: Aquest té com a finalitat crear una vista integrada de diverses fonts de dades, les quals són aparentment incompatibles. Aquesta incompatibilitat ve donada per les diferents percepcions i requeriments que sovint fan expressar de múltiples formes una mateixa informació.
- Data cleansing [16]: Aquí, l'objectiu és detectar i eliminar els errors i inconsistències en les dades, de manera que se'n millori la seva qualitat. Els més comuns poden ser de manca de dades, dades invàlides o paraules incompletes.
- Object integration [19]: Consisteix en establir relacions entre objectes que tenen propietats, o comportaments, similars.
- Master data management : L'objectiu és nodrir de processos que permetin recollir, agregar, creuar, consolidar i assegurar la qualitat, de la informació de les organitzacions per garantir-ne el control i permetre'n un ús controlat.

Com ja s'ha dit, el Record Linkage permet fer re-identificació i, a més, freqüentment és útil per a fer una neteja de les dades [23] i en la integració de conjunts de dades distribuïts i heterogenis. L'objectiu principal serà, doncs, acabar identificant quins registres dins d'un conjunt corresponen a una mateixa entitat física.

2.2 Procés de Record Linkage

El Record Linkage consta de diverses etapes [21], cadascuna d'elles encarregada d'una tasca i executada de forma seqüencial. A la Figura 2.1 podem veure un diagrama de les subdivisions que té aquest procés.

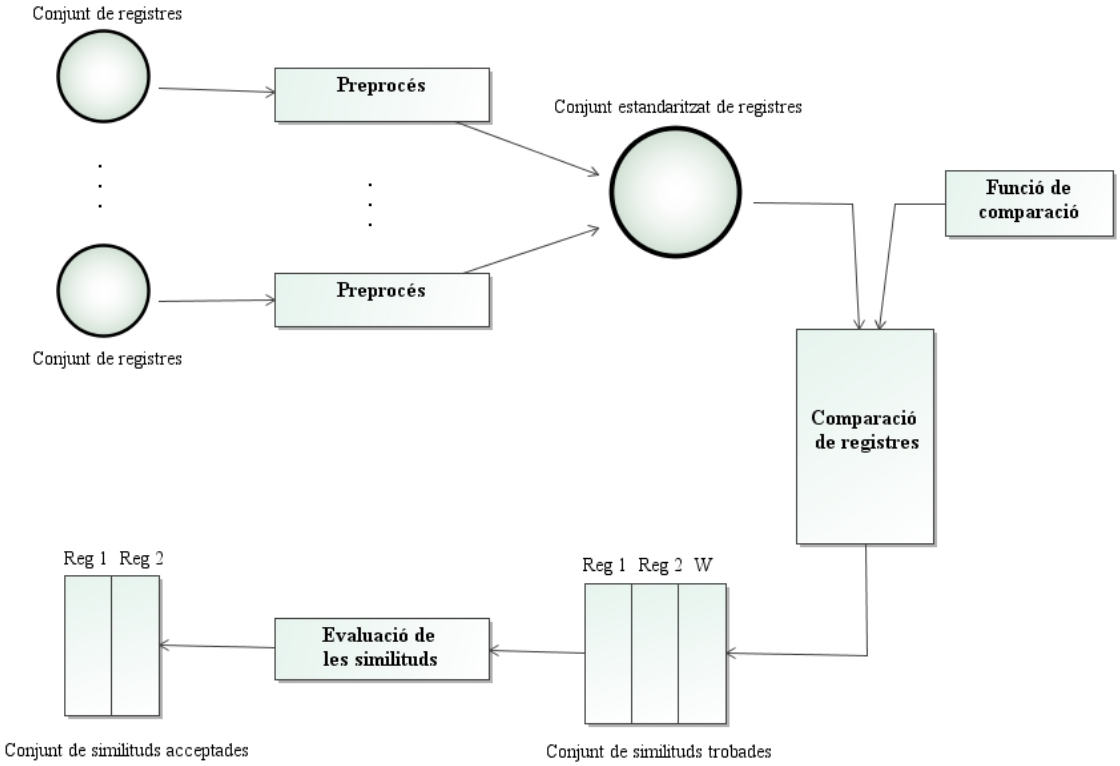


Figura 2.1 : Fluxe d'informació d'un procés de Record Linkage

En el món real, les fonts de dades solen contenir informació bruta com ara bé errors tipogràfics, ortogràfics o abreviacions de paraules, i és també habitual que hi hagi una mancança de uniformitat en el seu format (accents, espais, símbols de puntuació,...). És per això que la primera fase del procés, a la qual anomenem **preprocés**, és l'encarregada de normalitzar la informació de les diverses fonts de dades, netejar-la i unir-la en una única font.

La segona fase és la més costosa i té la funció de **comparar** tots els registres entre ells. Per a fer aquesta tasca, es fa ús d'una funció de comparació de registres, la qual servirà per a fer el càlcul del grau de semblança W. Donats dos registres x i y compostos per n atributs, es calcula W de la següent manera: $W(x,y) = \sum_{i=1}^{n} W_i(x_i, y_i)$, on x_i és l'atribut i del registre x. Així doncs, el grau de similitud d'un parell de registres, també anomenat **pes** d'una similitud, es calcula com la suma dels graus de similitud parcials entre els seus atributs comparats entre ells. El resultat que s'obté en finalitzar aquesta fase és un conjunt de parells de registres juntament amb el seu pes.

La tercera i última fase és l'**avaluació dels resultats** obtinguts en la segona fase. Una vegada es tenen les similituds trobades a partir de les comparacions entre registres, falta decidir quines es marquen com a realment idèntiques i quines són rebutjades. Aquesta tasca, doncs, es fa de forma manual, ja que s'ha de fer un anàlisi un a un dels resultats obtinguts.

Degut a que la primera i tercera fases no són imprescindibles, és habitual considerar Record Linkage només a la segona fase, la qual és totalment necessària si es volen trobar similituds entre registres. Per tant, quan ens referim a RL, ens estarem referint a l'algorisme de comparació de registres; a les altres dues fases les anomenarem **preprocés**

i **avaluació de resultats o revisió** respectivament.

Com ja s'ha comentat prèviament, la fase de comparació de registres és computacionalment molt costosa i el seu cost és quàdratic $O(N^2)$ respecte el nombre de registres. En canvi, les altres dues fases tenen un cost linial $O(N)$. És per això, que els esforços es solen centrar en reduir la complexitat que suposa la comparació un a un dels registres i és aquí on intervenen els anomenats mètodes de *blocking*.

2.3 Mètodes de blocking

En un procés de Record Linkage, la comparació de cada registre de les fonts de dades amb tots els altres és molt costosa a nivell computacional. És per això, que els mètodes de *blocking* són emprats, ja que redueixen considerablement aquesta problemàtica i per tant ajuden a no haver de comparar cada registre amb tots els altres.

Els mètodes de *blocking* utilitzen els atributs dels registres per a dividir les dades en blocs. Així, cada registre passa a formar part d'un bloc i només s'haurà de fer la comparació d'ell amb els altres registres que formen part del bloc al que pertany, amb la qual cosa es redueix considerablement el problema. Els dos mètodes clàssics són: **Standard Blocking** i **Sliding Window** , els quals explicarem a continuació.

2.3.1 Standard Blocking

El mètode Standard Blocking (en endavant SB) agrupa els registres en blocs mitjançant una **clau de bloc** o blocking, la qual es defineix a partir dels atributs dels registres. Un exemple de clau pot ser: *aquells registres tals que coincideixin en els cinc primers números de l'atribut Telèfon*. A més, en una clau poden intervenir més d'un atribut i per tant, podriem tenir una clau de bloc com: *aquells registres tals que coincideixin en els cinc primers números de l'atribut Telèfon i tinguin igual l'atribut Sexe*.

Així, a partir dels registres d'exemple de la Figura 1.1 i emprant com a clau *aquells registres tals que tinguin iguals els quatre primers caràcters de l'atribut Nom*, s'obtindrien sis blocs, tal i com es mostra a la Figura 2.2. Per tant, només serien necessàries dues comparacions de registres (les dels dos registres dels Bloc 3 i 5), mentre que si no empréssim el mètode SB s'haurien hagut de dur a terme vint-i-vuit comparacions per a poder comparar tots els registres entre ells.

Bloc 1	Jesep M.	Subirats	Puigfalcó	65	H	Barcelona
Bloc 2	Jesús	Santana	Masdeu	33	H	Arbúcies
Bloc 3	Josep M.	Subirats	Puigfalcó	65	H	Barcelona
	Josep Maria	Sovirats	Puigfalcó	65	H	Barcelona
Bloc 4	Roger	Subirats	Garcia	21	H	Girona
Bloc 5	Ramon	Solé	Recasens	83	D	Vic
	Ramona	Solé	Recasens	93	D	Vic
Bloc 6	Rosalia	Subirats	Gracia	23	D	Mataró

Figura 2.2 : Blocs obtinguts a partir d'aplicar Standard Blocking als registres de la Figura 1.1 utilizant la clau *aquells registres tals que tinguin iguals els quatre primers caràcters de l'atribut Nom*.

Aplicant claus de bloc, podem veure que el nombre de comparacions es pot reduir de forma substancial, ja que es passa de comparar tots els registres entre ells a comparar només els que formen part del mateix bloc. Així, si tenim N registres i el nombre de blocs obtinguts emprant SB és b (tots ells amb el mateix nombre de registres), es pot dir que la mida dels blocs és $B = N/b$. Per tant, la complexitat de l'algorisme passar a ser $O(B{\cdot}N)$. Aquest resultat s'obté en un cas ideal, el qual és pràcticament inassumible amb dades reals, ja que difícilment tots els blocs tindran la mateixa mida. És per això que se sol dir que la complexitat de SB ve dominada pel bloc que conté més registres i, per tant, és $O(B'{\cdot}N)$, on B' és la mida d'aquest bloc.

Una recomenació que s'ha de tenir en compte a l'hora d'escollir una clau és la d'utilitzar l'atribut que es consideri menys propens a tenir errors [13]. En l'exemple de la Figura 2.2, es pot apreciar que el registre del primer bloc correspon a la mateixa entitat física que els registres del tercer bloc i, per tant, s'estaran deixant de fer unes comparacions que tindrien com a conseqüència trobar finalment més similituds. Una solució a aquest problema és realitzar diferents **iteracions** d'aquest procés, emprant, a cada iteració, una clau de bloc diferent. D'aquesta forma, en cada repetició del procés, es generaran blocs diferents i permetrà la detecció de noves similituds. Com a conseqüència de realitzar aquestes iteracions serà possible que es detectin similituds ja trobades anteriorment, la qual cosa significarà que s'introdueixi una nova fase en el procés que anomenarem eliminació de similituds repetides.

2.3.2 Sliding Window

El mètode de Sliding Window (en endavant SW) ordena els registres segons una **clau d'ordenació** donada, la qual està composada com en el mètode de Standard Blocking per un o més atributs. Un cop feta la ordenació, es defineix una window o **finestra**, que tindrà un nombre màxim de registres amb els que es vol comparar cadascun d'ells. Així, la finestra w_i serà el conjunt de w registres que seran comparats amb el i-èssim registre i aquesta anirà desplaçant-se registre a registre tal i com es mostra a la Figura 2.3.

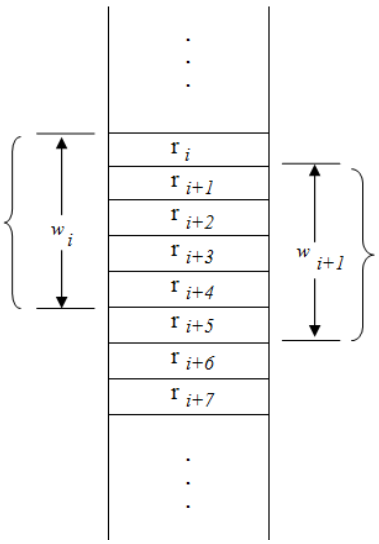


Figura 2.3 : Il·lustració del desplaçament de les finestres en el mètode SW amb una finestra $w = 5$.

Per tant, aquest mètode treballa a partir d'ordenar lexicogràficament cadascuna de les claus d'ordenació (atributs) per així comparar cada registre amb els següents w que el segueixen [24]. Així, és important definir el valor de w de tal manera que no sigui massa petit, ja que llavors es deixarien de comparar registres que podrien correspondre a la mateixa entitat, però tampoc massa gran, ja que implicaria estar fent comparacions de registres que s'assemblen molt

poc. També de gran importància és triar una clau d'ordenació que tingui una alta capacitat de discriminació i que sigui propensa a tenir pocs errors en les seves dades. A la Figura 2.4 es pot observar, a partir de les de l'exemple de la Figura 1.1, una il·lustració de com quedarien ordenades si la seva clau és l'atribut *Nom*; el nombre de comparacions que posteriorment es faran dependrà del valor de la finestra w.

Nom	Cognom 1	Cognom 2	Edat	Sexe	Població
Jesep M.	Subirats	Puigfalcó	65	H	Barcelona
Jesús	Santana	Masdeu	33	H	Arbúcies
Josep M.	Subirats	Puigfalcó	65	H	Barcelona
Josep Maria	Sovirats	Puigfalcó	65	H	Barcelona
Ramon	Solé	Recasens	83	D	Vic
Ramona	Solé	Recasens	93	D	Vic
Roger	Subirats	Garcia	21	H	Girona
Rosalia	Subirats	Gracia	23	D	Mataró

Figura 2.4 : Exemple d'ordenació dels registres de la Figura 1.1 mitjançant la clau *Nom*.

Com en el cas de SB, en aquest mètode també se li poden aplicar diverses **iteracions** amb claus diferents per a així trobar més possibles similituds. Això, de igual manera, suposarà introduir una darrera fase que servirà per a detectar similituds repetides degut a aquest procés de repetició. Aquesta fase l'anomenarem eliminació de similituds repetides.

El cost d'aquest segon mètode de *blocking* vindrà marcat pel valor que tingui la finestra i serà, per tant, $O(w \cdot N)$, on w és el valor d'aquesta finestra i N el nombre de registres . A més, el cost de la ordenació lexicogràfica serà de $O(N \cdot log(N))$ i el de la fase d'eliminació de similituds repetides serà linial $O(N)$. Per tant, el cos de Sliding Window serà el major entre la fase d'ordenació i la fase de comparació (dependrà de si w és major o no que $log(N)$).

2.4 Conclusions obtingudes

Es pot observar que els mètodes de *blocking* són molt útils per a reduir el nombre de comparacions que es duen a terme en un proces de Record Linkage i, per tant, se'n disminueix la seva complexitat computacional [13].

Per a millorar la precisió d'aquests mètodes, se solen fer diverses iteracions, fet que suposa també haver d'incloure una nova fase que anomenem eliminació de similituds repetides. Com a conseqüència, la fase de comparació de registres de la Figura 2.1 passa a estar formada per tres fases il·lustrades a la Figura 2.5.

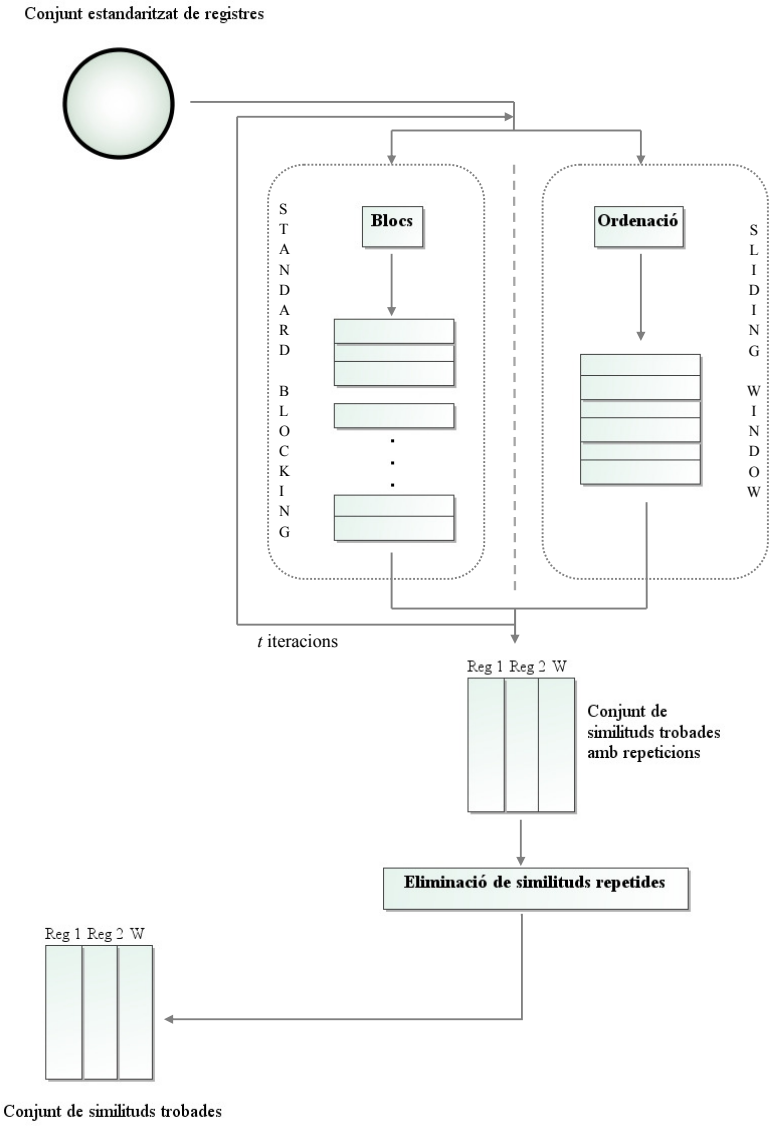


Figura 2.5 : Il·lustració de les diferents etapes que composen la fase de comparació de registres per als mètodes de *blocking*. A l'esquerra veiem el mètode de Standard Blocking i a la dreta el de Sliding Window. Ambdós comparteixen la última fase d'eliminació de similituds repetides com a conseqüència de les iteracions amb diferents claus.

Així doncs, el Record Linkage té una base sòlida i se n'ha fet i se n'està fent molta recerca per a optimitzar-ne els seus resultats i per a fer-ho de forma més ràpida aplicant noves tècniques.

Capítol 3

Visió general del projecte

Aquest tercer capítol té com a objectiu ampliar la descripció feta del projecte en el primer capítol aprofitant dels coneixements exposats en el segon. La idea és proporcionar una visió ràpida i clara de les funcionalitats, objectius fixats i resultats que es volen obtenir amb la construcció d'aquesta eina sense entrar en profunditat en aspectes que fan referència a l'àmbit informàtic.

3.1 Descripció

Cal recordar que el principal objectiu del sistema que es vol implementar és poder trobar aquelles entrades d'un conjunt de fonts de dades que corresponen a una mateixa entitat segons els criteris i restriccions que estableixi l'usuari. A més, es vol que tota la informació estigui centralitzada ja que, si no és així, és força freqüent que s'acabin tenint dades inconsistents.

Per a donar solucions a les mancances trobades en la versió prèvia s'ha decidit que es tractarà d'una aplicació web. Això significarà que els usuaris podran utilitzar-la accedint amb el seu navegador a un servidor web per mitjà d'internet o una intranet. Així, la informació quedarà emmagatzemada de forma centralitzada en el servidor i no es podran tenir problemes d'inconsistència entre dades de diferents usuaris, ja que tots ells estaran utilitzant les mateixes.

Un altre punt important serà que totes les dades i la informació necessària per al funcionament de l'aplicació estaran emmagatzemades en una base de dades. A més, existirà una auditoria, la qual suposarà tenir constància sobre cada modificació que facin els usuaris sobre la base de dades i aportarà un major control sobre la informació emmagatzemada, ja que tot canvi quedarà registrat en una taula junt amb el seu autor.

A la Figura 3.1 es mostra un esquema de com es farà ús de l'aplicació.

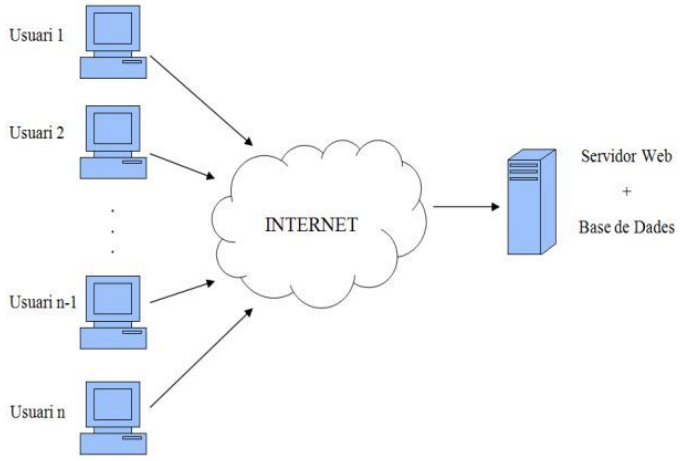


Figura 3.1 : Diagrama del funcionament d'una aplicació web.

A més, s'introduirà la gestió d'usuaris, de manera que tothom que vulgui utilitzar l'aplicació haurà d'autenticar-se per a poder accedir-hi. Es permetrà que les tasques es facin de forma col·laborativa, ja que cada usuari es podrà especialitzar en una part del procés de RL i tots els altres es beneficiaran de la feina feta. S'incorporarà la possibilitat de que la tasca de revisió de similituds es pugui realitzar de forma conjunta entre varis usuaris, havent-lis d'assignar prèviament un rang a cadascun d'ells, aconseguint així que aquesta avaluació es pugui fer de forma més ràpida i eficient.

Com a qualsevol aplicació, es distingiran diferents rols d'usuari, cadascun d'ells amb permisos diferents (*vegeu Capítol 3.3*). També es dotarà cadascuna de les pàgines que composen l'aplicació d'una ajuda pels usuaris i es d'un assistent per a poder fer un procés guiat del Record Linkage.

El principal element d'aquest sistema seran les **fonts de dades**. Aquestes podran ser afegides al sistema per part dels usuaris amb l'objectiu d'aplicar-lis el Record Linkage i també podran esborrar-se quan ja no siguin utilitzades. El format d'aquests fitxers haurà de ser inicialment en CSV (*Comma Separated Value*) i tots ells quedaran emmagatzemats al servidor, de manera que tots els altres usuaris també puguin utilitzar-los.

Tal i com s'ha introduit en el Capítol 2.2, el procés de RL es caracteritza per estar dividit en tres fases: el **preprocés** de dades, la **comparació** de registres i l'**avaluació** dels resultats obtinguts. És per això, que la principal tasca per l'usuari serà parametritzar les dues primeres fases per a obtenir uns resultats que podran avaluar-se a la tercera fase.

Una vegada introduides les fonts de dades, es podrà iniciar el **preprocés** d'aquestes, en el qual s'estandaritzarà la informació per a obtenir uns millors resultats i s'escollirà quines columnes es volen tenir en compte. Més endavant es mostraran més detalls de com es pot estandaritzar i netejar aquesta informació (*vegeu Capítol 3.2)*.

Per últim, se seleccionarà l'estratègia que es vol utilitzar (**algorisme**) per a fer la comparació de registres (*vegeu Capítol 2.3*). En aquest PFC s'ha decidit donar la possibilitat d'utilitzar un sol algorisme, el de *Sliding Window*, de manera que s'hauran d'escollir els paràmetres que caracterizen aquest mètode: mida de finestra i claus d'ordenació.

Aquestes tres tasques (fonts de dades, preprocés i algorisme) estaran englobades en mòduls diferents, de manera que podran realitzar-se de forma independent, sense haver-les de realitzar totes a la vegada. Tot i així, existiran dependències de dades entre elles, el qual significarà que no podrà dur-se a terme una sense haver-ne fet l'anterior (més informació al *Capítol 3.2)*.

Finalitzades aquestes tasques, es procedirà a l'execució de les comparacions entre registres, a la que habitualment anomenarem **execució**, amb la qual s'obtindrà com a resultat un conjunt de parelles de registres amb un grau de similitud (**similituds**), les quals podran assignar-se a un rang d'usuaris per a que les revisin. Degut a que els resultats

obtinguts poden ser de dimensions considerables i, per tant, la fase d'**avaluació** pot ser la més costosa, és necessari poder visualitzar tota la informació de manera clara, entenedora i amigable. És per aquest motiu que les similituds es mostraran en el que anomenem **graella de revisió**, la qual permetrà visualitzar un conjunt de similituds a la vegada en comptes d'una de sola i donarà l'opció de fer una revisió conjunta o massiva.

Per finalitzar, es donarà la possibilitat d'exportar les dades. Aquesta funcionalitat també haurà de ser editable per a així donar la opció a l'usuari de que escolleixi aquelles dades, en diferents formats i criteris, que siguin d'interés sobre les similtuds trobades.

A la Figura 3.2 es presenta de forma esquematitzada tot el procés descrit anteriorment, sense entrar en detall sobre com es farà cadascuna de les fases.

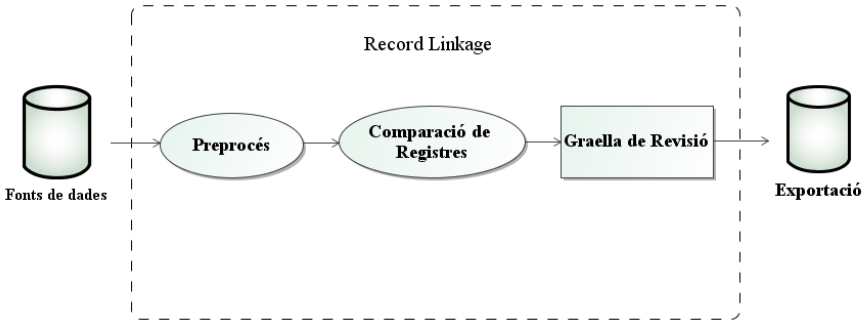


Figura 3.2 : Esquema del curs de l'aplicació.

3.2 Flux de treball

Després d'haver descrit les funcionalitats que conformen el sistema software a desenvolupar, és el moment d'entrar a fons a cadascuna d'elles i descobrir quins elements hi intervenen.

La característica més important és que l'aplicació treballa en un entorn de configuracions, les quals serviran per a definir els diferents paràmetres necessaris en un procés de Record Linkage. Així, cada punt dels esquematitzats a la Figura 3.2 coincidirà amb una configuració. Les configuracions seran les següents: **configuració de fonts de dades**, **configuració de columnes de creuament**, **configuració de l'algorisme** i **configuració d'exportació**. Aquestes podran crear-se per separat dins de l'aplicació, però existiran relacions entre elles, tal i com mostra el diagrama de dependències de la Figura 3.3. Com es pot observar, la creació d'una configuració de l'algorisme dependrà d'haver-ne creat d'una de columnes de creuament i així amb totes les dependències que es veuen a la figura.

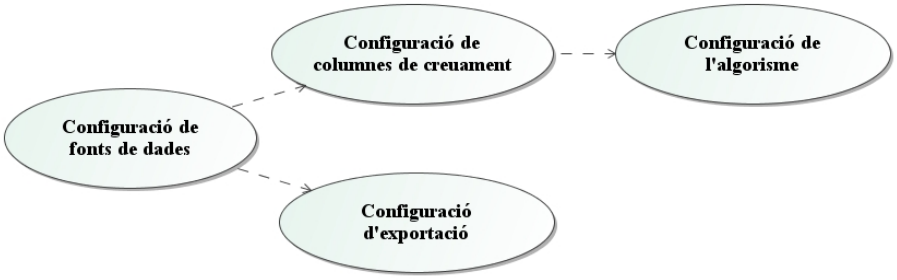


Figura 3.3 : Diagrama de dependències entre configuracions de Daurum. Per a crear-ne una d'elles, s'ha d'escollir abans sobre quina de les seves depèndencies prèvies es vol aplicar.

Degut a l'existència d'aquestes configuracions, existirà doncs una **gestió de configuracions**, en la qual es podràn afegir-ne de noves i esborrar-ne. A continuació, es descriu en detall la funcionalitat de cadascuna d'elles.

Configuració de fonts de dades

La configuració de fonts de dades és la que permetrà als usuaris pujar les dades a les que es vol aplicar un procés de RL. Es permetran afegir tants fitxers com es vulguin i es mostrarà una previsualització d'ells per a que l'usuari pugui assegurar-se de que realment contenen les dades desitjades. Els fitxers amb format CSV es caracteritzen per tenir un caràcter separador, el qual serveix per a separar les columnes o atributs dels registres (files).Les Figures 3.4 i 3.5 mostren dues interpretacions de les columnes d'un mateix fitxer de dades d'aquest tipus utilitzant dos caràcters separadors diferents.

c0	c1	c2
0-org	arroyo gasco	00000008A
1-org	rosa maria ballester navarro	00000001A
2-org	manuel calvet gomez	00000002A
3-org	carmen manso gonzalez	00000003A
4-org	eva guillen llopart	00000004A
5-org	ana maria campos joya	00000005A
6-org	victoria pozuelo bassas	00000006A
7-org	marcos navarro	00000007A
8-org	immaculada vilanova carrasco	00000008A

Figura 3.4 : Dades amb ';' com a caràcter separador.

c0	c1
0	org; arroyo gasco;00000008A
1	org;rosa maria ballester navarro;00000001A
2	org;manuel calvet gomez;00000002A
3	org;carmen manso gonzalez;00000003A
4	org;eva guillen llopart;00000004A
5	org;ana maria campos joya;00000005A
6	org;victoria pozuelo bassas;00000006A
7	org; marcos navarro;00000007A
8	org;immaculada vilanova carrasco;00000008A

Figura 3.5 : Dades amb '-' com a caràcter separador.

Configuració de columnes de creuament

La configuració de columnes de creuament és necessària per a estandaritzar i netejar les dades introduïdes a la configuració anterior i depèn d'haver creat prèviament una configuració de fonts de dades. El punt més important aquí és l'elecció de les anomenades **columnes de creuament**, que serviran per a indicar quines columnes de les fonts d'entrada es voldran fer servir per a aplicar el procés de RL i també per a relacionar les columnes de les diferents fonts entre elles. Així, aquestes columnes seran les que es faran servir per a aplicar Record Linkage, mentre que les restants columnes dels fitxers d'entrada no es faran servir, degut a que no aportaran informació per a trobar duplicats. A més, es podran posar pesos a les columnes de creuament escollides per a així donar-lis mes o menys importància a la hora de comparar els registres. Aquesta configuració tindrà un elemet visual força treballat per a que l'usuari pugui saber en tot moment quina és cadascuna de les columnes que està escollint i quina informació té associada veient una previsualització dels seus valors.

Un exemple sobre l'ús d'aquest tipus de columnes és el següent: donats un fitxer d'entrada *A* amb columnes *Nom, Cognom1, Cognom2* i *Població* i un fitxer *B* amb columnes *Cognom1, Cognom2* i *DNI* definirem dues columnes de creuament, degut a que només existeixen dues columnes a cada fitxer que continguin la mateixa informació: el Cognom1 i el Cognom2. Així, la primera columna de creuament estarà formada per la segona columna del fitxer *A* i la primera columna del fitxer *B* i correspondrà amb el Cognom1; mentre que la segona columna de creuament correspondrà amb el Cognom2 i la composaran la tercera columna del fitxer *A* i la segona del fitxer *B*. A la Figura 3.6 es pot observar gràficament l'exemple explicat.

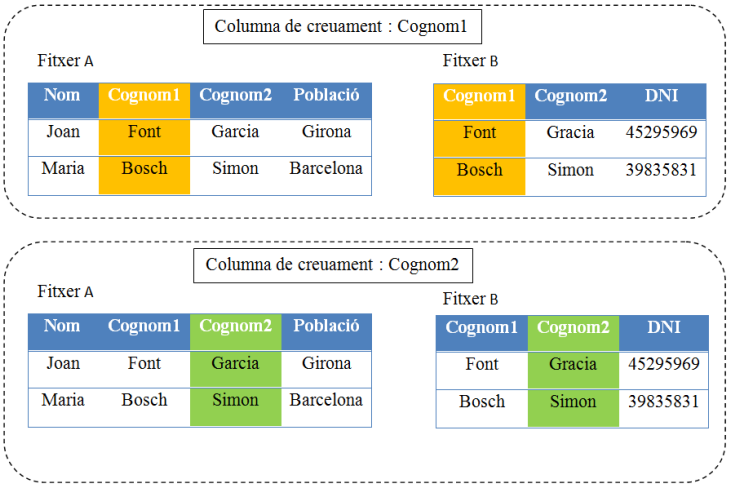


Figura 3.6 : Exemple de dues columnes de creuament amb dos fitxers d'entrada *A* i *B*. Una columna de creuament està composada per les columnes de color taronja i l'altra per les de color verd.

Per últim, cal comentar també que les columnes que composen una columna de creuament podran filtrar-se per a transformar els seus valors. Així, es podran unificar formats de dates, esborrar accents de paraules o fins i tot substituir paraules per unes altres. A continuació es mostran exemples de filtratge: passar de *Lluís* a *LUIS*, substituir *Pep* per *Francesc* o passar la data *22/1/09* a *22/01/2009*. Aquests filtratges es faran mitjançant uns filtres editables pels usuaris i, per tant, apareixerà una nova funcionalitat que serà la de **gestió de filtres**. Per a editar-los, la mateixa aplicació proporcionarà un entorn on es resaltarà el seu codi, el qual haurà de ser en *JavaScript*, i es podrà provar in situ el seu funcionament.

Configuració d'algorisme

La configuració de l'algorisme és la destinada a contenir la informació útil per al mètode de comparació de registres que s'utilitzarà, que serà el de Sliding Window. Així, aquí l'usuari definirà el tamany de la finestra i les claus d'ordenació, que anomenarem **columnes d'ordenació** i també sobre quina configuració de columnes de creuament es vol aplicar. Les columnes de creuament hauran d'estar formades per un subconjunt de les columnes de creuament definides amb anterioritat.

Execució

Les tres configuracions anteriorment definides, la de **fonts de dades**, la de **columnes de creuament** i la de l'**algorisme** seran les necessàries per a poder executar el procés de Record Linkage. Així doncs, a part de tenir la gestió de les configuracions, es tindrà una funcionalitat principal que serà l'**execució**, que dependrà d'haver creat prèviament les configuracions adequades. Durant l'execució, s'anirà informant en tot moment a l'usuari de l'estat en que es troba i donarà mètriques sobre el temps que trigarà en realitzar-se. Una vegada acabat, es donarà informació sobre els resultats obtinguts i es permetrà assignar la seva revisió a un rang d'usuaris.

Graella de revisió

La graella de revisió és la eina que servirà als revisors per a decidir l'estat de les similituds trobades: acceptada, rebutjada o per determinar. Així, és important que aquest element sigui fàcilment manegable i intuitiu per a l'usuari.

Es tractarà d'una graella que contindrà varies pàgines, on a cada pàgina hi haurà un conjunt de similituds a revisar i on l'usuari podrà fàcilment canviar l'estat d'elles. A més, incorporarà la opció de **revisió massiva** segons els graus de semblança i també dues maneres de visualitzar les similituds: **vista per similituds** i **vista per grups**. La primera vista provindrà d'ordenar descendentment el grau de semblança i la segona d'agrupar aquelles similituds que comparteixen registres.

Així doncs, apareix un element, al qual anomenem **grup**. Un grup estarà compost com a mínim per una similitud i serà una manera d'ajuntar totes les similituds que potencialment podrien correspondre a una sola entitat física. Això significa que existeix un lligam entre elles degut a que coincideixen en un dels registres de les similituds. Una manera gràfica d'expressar el que és un grup és pensant en grafs. Representarem a tots els registres als quals se'ls hi ha trobat una similitud amb nodes i les similituds trobades entre ells amb les arestes, de manera que entre un registre *r1* i un registre *r2* hi haurà aresta si l'execució del mètode de Record Linkage ha trobat una similitud entre ells. Així, hi haurà tants grups com components connexes tingui el graf, tal i com mostra la Figura 3.7.

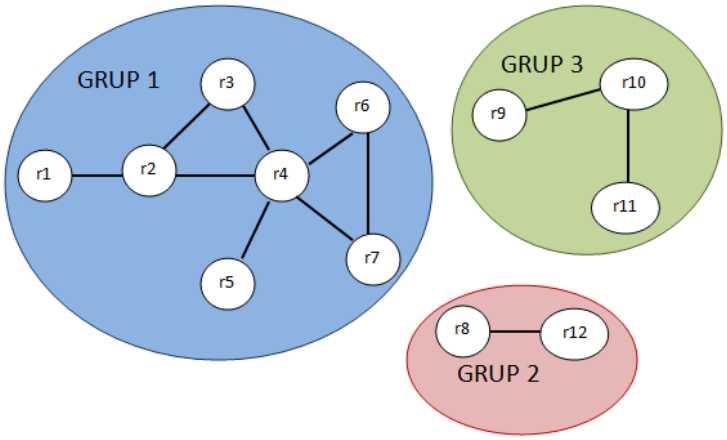


Figura 3.7 : Explicació dels grups de similituds en forma de graf. Existeixen tres grups que corresponen amb els tres components connexes del graf.

A la Figura 3.8, veiem les similituds obtinguts d'executar les dades de la Figura 3.6 amb ambdues columnes de creuament com a claus d'ordenació. Es pot observar com ha trobat dos registres que s'assemblen en un 100% i per tant ha estat acceptada, mentre que hi ha altres similituds que no s'assemblen gens i per tant tenen un 0% de pes i han estat rebutjades. Per últim, comentar també que els valors de les columnes de creuament (*Cognom1* i *Cognom2*) han estat filtrats mitjançant un filtre que passa les lletres a majúscules i treu accents i altres símbols poc comuns, degut a que així s'ha escollit a la configuració de columnes de creuament.

Id	State	Weight	Cognom1	Cognom2	Registers
0	✓	100.0	BOSCH BOSCH	SIMON SIMON	Maria;Bosch;Simon;Barcelona Bosch;Simon;39835831
1	⚠	98.95	FONT FONT	GARCIA GRACIA	Joan;Font;Garcia;Girona Font;Gracia;45295969
2	✗	0.0	BOSCH FONT	SIMON GRACIA	Maria;Bosch;Simon;Barcelona Font;Gracia;45295969
3	✗	0.0	FONT BOSCH	GARCIA SIMON	Joan;Font;Garcia;Girona Bosch;Simon;39835831

Figura 3.8 : Exemple de graella de revisió amb quatre similituds trobades.

Configuració d'exportació

La configuració d'exportació és l'encarregada de definir el format de sortida dels resultats d'un procés de Record Linkage. Amb la finalitat de que l'usuari pugui exportar les dades que més convinguin, aquesta configuració estarà formada per un conjunt de **columnes d'exportació**. Una columna d'exportació es defineix de la mateixa forma que una de creuament, amb la diferència de que no fa falta triar exactament una columna de cada fitxer de dades. El seu significat serà el següent: per a una similitud en concret, el valor de cada columna d'exportació serà un d'entre el rang triat de columnes i l'usuari podrà triar-ne quin. Així, per a crear una configuració d'aquest tipus, s'ha d'escollir sobre quina configuració de fonts de dades es vol aplicar. Una vegada definida, es podrà dur a terme l'**exportació**, decidint l'usuari quin valor exporta per a cada similitud. Els formats en els que es podrà exportar seran: PDF, XLS, XML i CSV.

La Figura 3.9 mostra un esquema del seu funcionament. Hi ha tres columnes d'exportació definides i es mostren els possibles valors que podrien tenir aquestes columnes per a l'exportació d'una similitud trobada de la Figura 3.8.

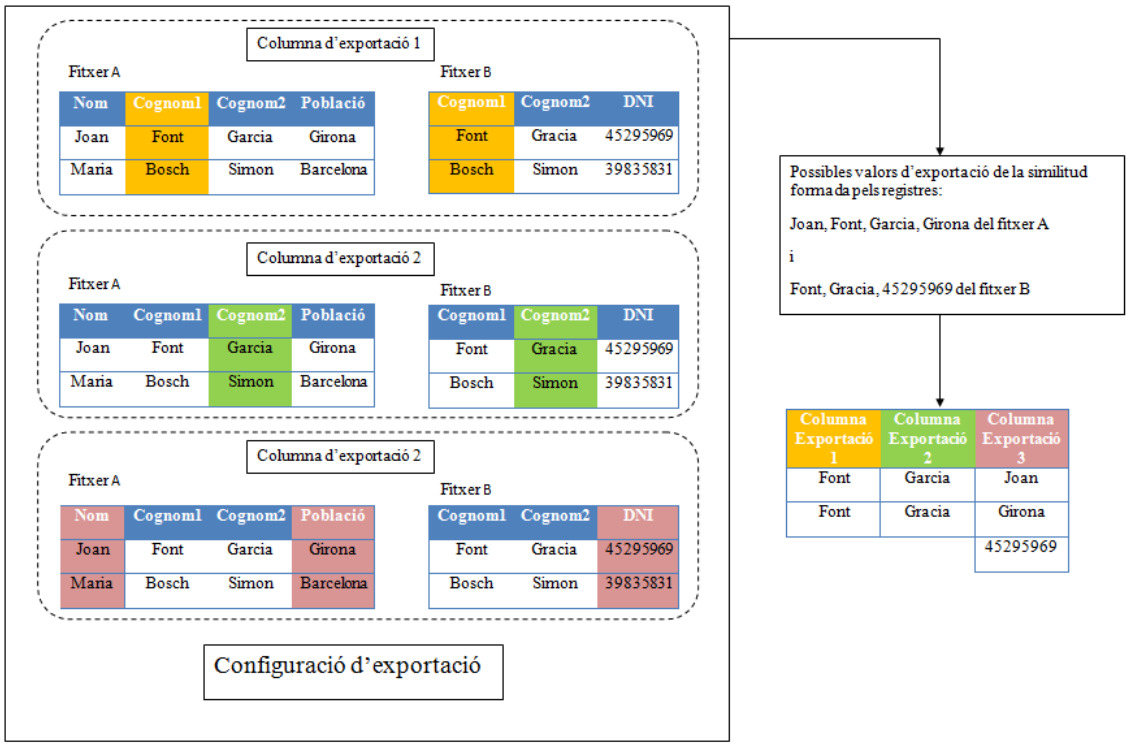


Figura 3.9: A l'esquerra es representa una possible configuració d'exportació amb tres columnes i a la dreta es mostra una taula amb els possibles valors d'una certa similitud. Per a les columnes 1 i 2 hi ha dos valors possibles mentre que per la columna 3 n'hi ha tres.

Resum

Així doncs, les funcionalitats fonamentals d'aquest projecte són: **gestió de les configuracions** de fonts de dades, de columnes de creuament, de l'algorisme i d'exportació, l'**execució** de RL, la **revisió** dels resultats, l'**exportació**, la **gestió de filtres** i, lògicament, la **gestió d'usuaris**. Tot plegat funcionarà sota una base de dades en la qual s'emmagatzemaran tota la informació descrita i on, a més, es farà un control en forma d'auditoria de tots els canvis que es produeixin. A la Figura 3.10 es pot veure un resum de tots els elements i tasques i les interrelacions que tindran aquestes amb els usuaris de l'aplicació.

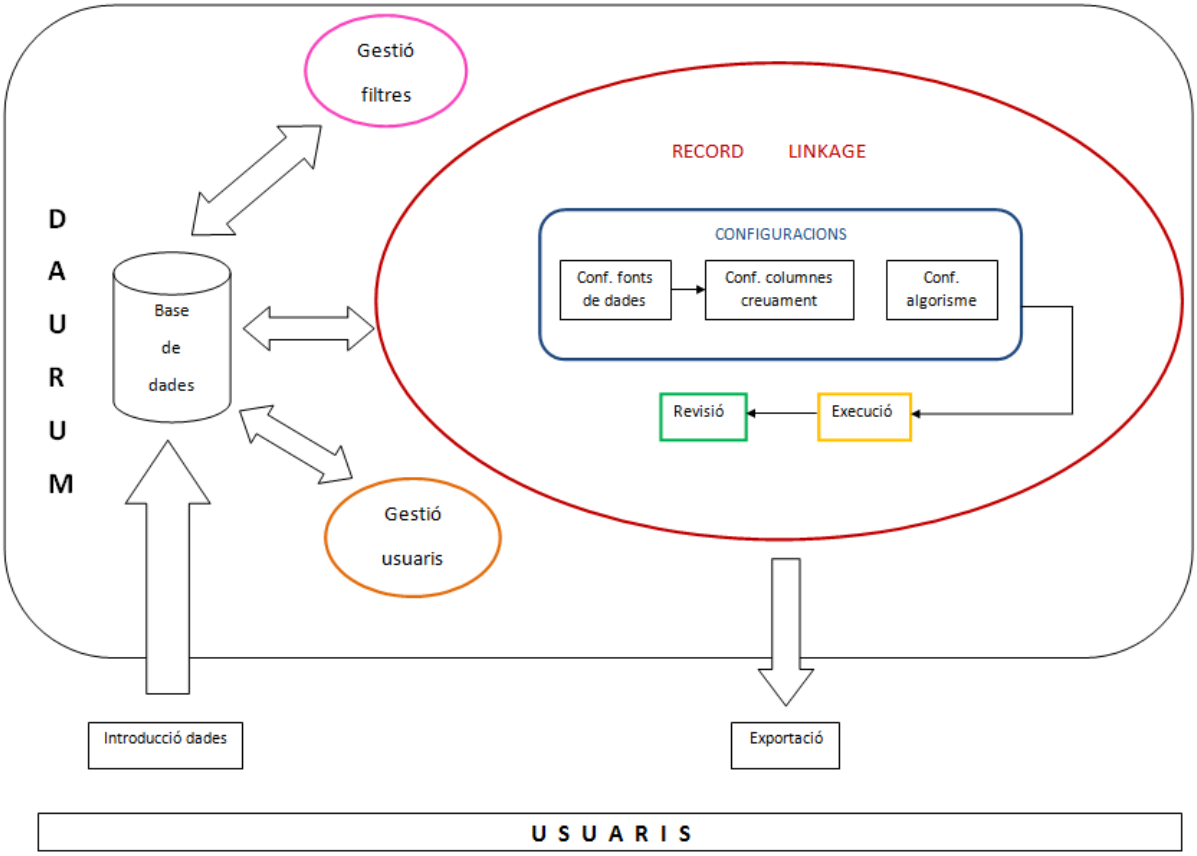


Figura 3.10: Resum de les interrelacions que existiran a Daurum.

Podem observar que el punt central de l'aplicació és la Base de Dades. Dins d'ella es guardarà tota la informació que formi part del sistema, de manera que quedarà persistent i sempre accesible per part dels usuaris. Existiran dos fluxos de dades entre Daurum i els usuaris que són la introducció de dades i l'exportació de resultats en els quals serà necessària l'existència d'arxius diversos: els CSV d'entrada des dels usuaris cap al sistema i el fitxers d'exportació (CSV,XML,XLS o PDF) des del sistema cap als usuaris. La part central de les funcionalitats serà la que engloba tot el relacionat amb el Record Linkage: les configuracions, l'execució i la revisió.

3.3 Usuaris

Els usuaris del sistema podran tenir diferents rols, els quals donaran el privilegi d'accedir a unes o altres funcionalitats. Així, els rols que existiran seran: **administrador**, **expert** i **revisor**. Degut a que ja coneixem les diferents tasques que es podran dur a terme, a continuació es procedirà a descriure quines d'aquestes pot utilitzar cadascun dels rols.

L'**administrador** serà encarregat de la gestió d'usuaris. Això significa que serà ell qui pugui afegir nous usuaris al sistema, així com esborrar-ne o editar-ne. Per tant, també serà qui els otorgui privilegis, donat que podrà editar els rols d'ells. És per això, que en tot moment haurà d'existir com a mínim un administrador al sistema i aquest no permetrà que se n'esborri l'últim en cas de que només n'hi hagi un.

L'**expert** serà l'encarregat tant de la gestió de configuracions (fonts de dades, columnes de creuaments i algorisme) com de l'execució del Record Linkage. Com el seu nom indica, es tractarà d'un gran coneixador del funcionament de RL, de manera que sabrà definir els paràmetres de les diferents configuracions per a un correcte funcionament de l'algorisme. A més, degut a que serà l'encarregat d'executar-lo, posteriorment també serà qui assigni les similituds resultants entre els diferents revisors. Per últim, també tindrà accés a la gestió de filtres, degut a que aquests es fan servir durant la configuració de columnes de creuament i, per tant, serà l'expert qui sàpiga què es vol filtrar. A més, l'expert també podrà accedir a exportar els resultats d'una execució ja revisada, i gestionar les configuracions d'exportació.

El **revisor** serà, com el seu nom indica, l'encarregat de revisar les similituds resultants de l'execució de Record Linkage. Es tractarà d'un gran coneixador en les dades d'entrada per a així poder decidir l'estat de les similituds. També tindrà participació en l'exportació dels resultats, degut a que escollirà ell quins valors s'exporten per a cada similitud.

Per últim, cal afegir també que tot usuari d'unes funcionalitats bàsiques independentment del rol que tinguin que són les següents: entrar al sistema (*login*), sortir del sistema (*logout*), canviar la seva contrasenya, consultar l'*about* i consultar l'ajuda existent.

Capítol 4

Especificació

En tot desenvolupament d'una aplicació és necessari realitzar un seguiment del cicle de vida de la seva construcció. Aquest capítol engloba la informació que fa referència a l'especifiació del sistema. Es defineixen els requisits funcionals i no funcionals del programa, es presenta un model de dades juntament amb una descripció detallada del funcionament i desenvolupament de les funcionalitats i es plasma de forma rigurosa com haurà d'estar construït aquest software per tal de facilitar-ne els posteriors disseny i codificació.

4.1 Introducció

Primer de tot, cal comentar que per a fer l'anàlisi, especificació i disseny de tot el sistema s'ha emprat l'estàndar UML atesa la seva llegibilitat i claredat, a més de la seva extensa utilització en el modelat de software actualment.

Una de les etapes més importants per a desenvolupar un software és la d'especificació d'aquest, ja que permet definir l'abast general per tal de detallar i documentar les funcionalitats que hauran de ser implementades en la futura codificació del software final. Aquesta etapa, inclou l'anàlisi de requisits i la creació d'un model de casos d'ús i conceptual del sistema.

En la enginyeria, existeixen molts models de desenvolupament de software que defineixen el cicle de vida que tindrà la seva construcció, com el model *clàssic* o *en cascada*, el qual defensa que totes les fases de la construcció s'ha de fer de manera seqüencial. En aquest projecte no el farem servir, ja que es tracta d'un model massa rígid i en el qual el software no pot anar evolucionant. És per això, que s'ha seguit un model en espiral, en el que es van fent diverses iteracions sobre les diferents fases que el composen per anar engrossant mica en mica les funcionalitats disponibles en el sistema.

A la Figura 4.1 veiem un esquema d'aquest model, en el qual la fase de *Determinar objectius* i *Anàlisi del risc* correspondran amb l'especificació i anàlisi de les noves funcionalitats que es desenvoluparan en una iteració concreta, la fase de *Desenvolupar i provar* amb la de disseny, implementació i testing de l'aplicació i la de *Planificació* amb una fase en la que es concretaran les següents funcionalitats a desenvolupar.

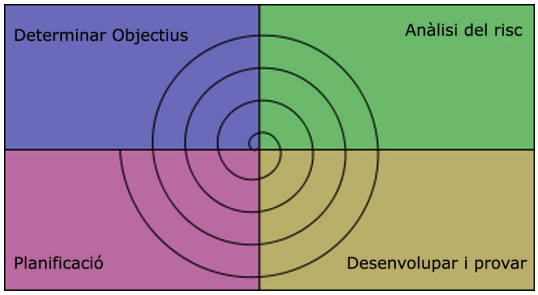


Figura 4.1: Esquema de la construcció del software

Una vegada detallat el procés que s'ha seguit per a la construcció de l'aplicació, es procedirà a mostrar els models de casos d'ús i conceptual. Abans però, s'ha hagut de fer un anàlisi de requisits, del qual sortirà una llista de requisits funcionals que coincidirà amb els diferents casos d'us que més endavant es mostren (*vegeu Capítol 4.3*).

4.2 Requisits no funcionals

En aquest punt, es llistaran el conjunt de requisits que no formen part de les funcionalitats del sistema, però que també és important tenir en compte.

- Interfície d'usuari: Els actors que interaccionin amb el sistema no han de ser necessàriament experts en l'àmbit de la informàtica. És per això, que s'haurà de tractar d'una eina fàcil d'utilitzar i molt intuïtiva. A més, degut a l'existència de diferents perfils d'usuari, s'haurà de fer distinció entre ells pel que fa als permisos d'accés de les diferents funcionalitats. Pel que fa al tipus de perifèrics d'entrada i sortida, únicament és necessari disposar d'una pantalla, un teclat i un ratolí per tal d'interaccionar amb el programa software.
- Ajuda: Tota aplicació necessita d'una bona ajuda documentada per tal de fer més entenedor el conjunt de funcionalitats del sistema i, d'aquesta forma, garantir-ne el màxim aprofitament. Es tindrà, doncs, ajuda sobre les diferents opcions de l'aplicació i, pel que fa al Record Linkage, hi haurà un procés guiat.
- Consideracions de software i hardware: Al tractar-se d'una aplicació web, els usuaris tan sols hauran de tenir accés a internet o una intranet i un navegador que permeti *JavaScript*. Pel que fa al servidor, s'ha decidit construir el sistema amb el llenguatge de programació Java [9] i, per tant, es podrà executar tant sobre un sistema operatiu *Microsoft Windows* com *Linux* i s'haurà de tenir instal·lat un servidor web que accepti *JSP* com pot ser *Tomcat* [6].
- Entorn de desenvolupament: Per a desenvolupar el sistema es farà ús d'un *IDE* potent com és *Eclipse* [8], el qual posseeix una gran quantitat de plugins que permeten afegir funcionalitats. A més, per a la gestió del projecte i les seves versions s'utilitzarà l'eina de *Maven* [3], que facilitarà una feina que a vegades pot provocar molts problemes de difícil solució com és la de resoldre les dependències existents entre diversos projectes.

4.3 Model de casos d'ús

En l'especificació d'un sistema informàtic no hi pot faltar la descripció detallada de cada cas d'ús. Cada cas, proporciona un curs d'esdeveniments típic que representa la interacció entre el sistema i l'usuari amb la finalitat d'assolir els objectius plantejats. Per tant, són utilitzats per a definir una seqüència d'accions que van succeïnt entre el sistema i l'usuari en resposta d'un esdeveniment iniciat per qualsevol d'aquests dos.

4.3.1 Diagrama de casos d'ús

Abans de descriure un a un tots els casos d'ús que conformaran l'aplicació, és necessari tenir una visió global d'aquests. Per a això, es mostrarà una representació en forma de diagrama per a veure quines interaccions hi haurà entre el sistema i els usuaris. En aquest projecte existiran tres tipus de usuaris diferents i, per tant, és d'ajuda veure cadascun d'ells a quins casos d'ús podrà accedir. Així, existiran tres diagrames diferents, els quals seran per un usuari **administrador**, un **expert** i un **revisor**. A més, hi haurà un quart, en el que s'inclouran els casos que poden ser utilitzats per tots ells.

Per a facilitar-ne la lectura de tots ells, s'ha decidit agrupar els casos d'ús en un nivell superior de casos dús genèrics, els quals coïncideixen amb els sis grans punts que s'ha comentat al Capítol 3.2 que tindria aquest sistema: la **gestió de configuracions**, l'**execució**, la **revisió**, l'**exportació**, la **gestió de filtres** i la **gestió d'usuaris**.

A la Figura 4.2 es pot observar la representació dels casos comuns amb tots els rols d'usuari diferents. A les posteriors, la Figura 4.3, la Figura 4.4 i la Figura 4.5 s'il·lustren els casos d'ús corresponents a l'administrador, el revisor i l'expert respectivament.

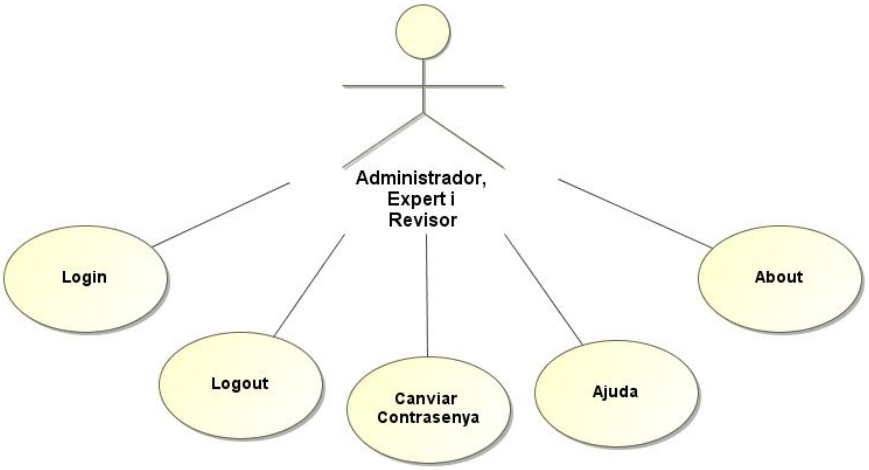


Figura 4.2: Diagrama dels casos d'ús compartits per tots els rols d'usuari

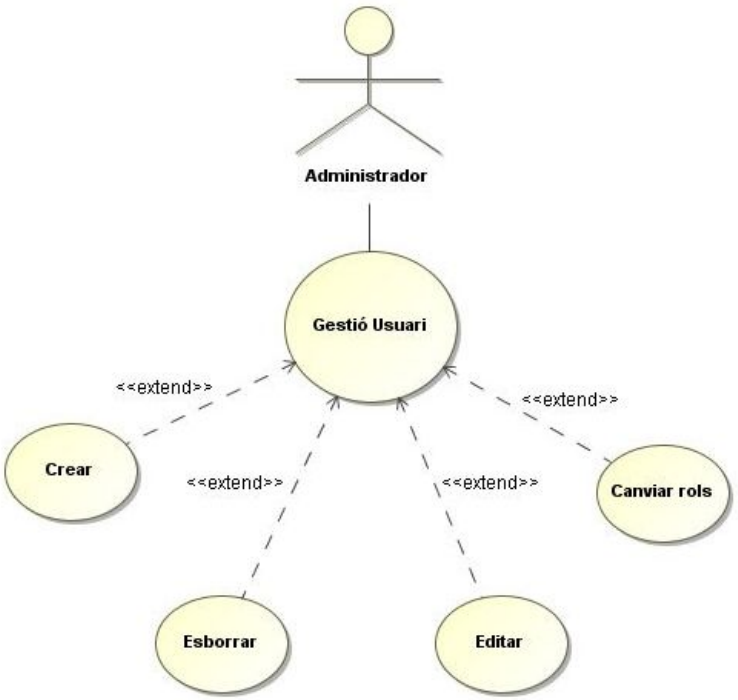


Figura 4.3: Diagrama dels casos d'ús de l'administrador

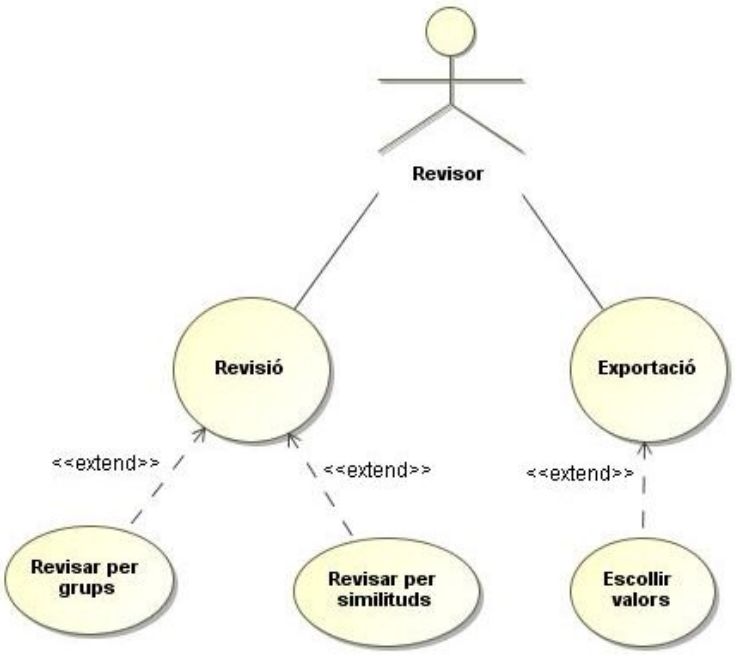


Figura 4.4: Diagrama dels casos d'ús del revisor

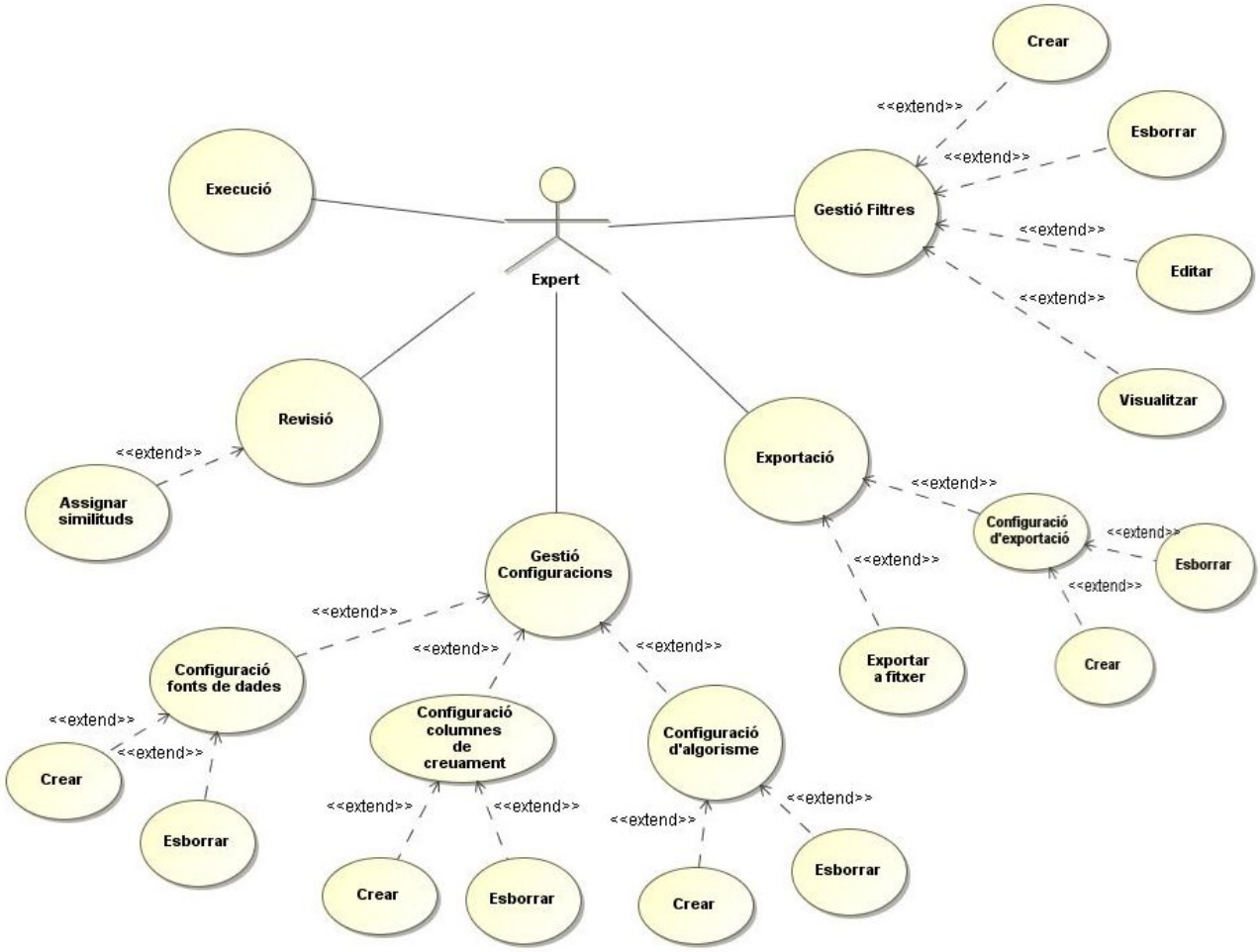


Figura 4.5: Diagrama dels casos d'ús de l'expert

4.3.2 Descripció de casos d'ús

Després d'haver donat una primera visió del conjunt de casos d'ús que tindrà el sistema mitjançant els diversos diagrames corresponents als diferents rols que existen, és el moment de descriure'n detalladament la seqüència d'accions que es duran a terme entre els actors i el sistema.

Els casos d'ús genèrics serviran per a llistar els elements existents al sistema del tipus desitjat: el cas d'ús gestió d'usuaris llistarà els usuaris existents, el de gestió de filtres mostrarà els filtres i així amb tots els altres casos. És per això, que s'ha decidit no descriure el seu comportament i només es definirà el de les seves extensions (funcionalitats típiques de gestió com són crear, esborrar, ...). A més, aquests casos d'ús genèrics ens serviran com a eina d'agrupament per així distingir millor les diferents funcionalitats existents al sistema. Cal dir que, degut a l'existència de diferents actors, s'especificarà en cada descripció de cas d'ús quin d'ells hi participa i, a més, s'inclouran quan convingui cursos alternatius al típic.

Casos d'ús comuns

Cas d'ús	Login
Resum	Permet accedir al sistema a un usuari
Actor	Administrador, Expert i Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari de fora del sistema decideix accedir a ell.	
	2. El sistema mostra un formulari de login amb els camps de nom d'usuari i contrasenya.
3. L'usuari omple els dos camps.	
	4. El sistema comprova que existeixi l'usuari amb la contrasenya entrats i, en cas afirmatiu, mostra nomes aquells menús i pàgina d'inici als quals tinguin accés els rols que posseeixi l'usuari.
Curs alternatiu	
4. El sistema detecta que els camps no són vàlids i mostra un missatge d'error.	

Cas d'ús	Logout
Resum	Permet sortir del sistema a un usuari
Actor	Administrador, Expert i Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari registrat de dins del sistema decideix sortir a ell.	
	2. El sistema treu a l'usuari de la sessió i mostra el formulari de login.

Cas d'ús	Canviar contrasenya
Resum	Permet canviar la contrasenya d'un usuari
Actor	Administrador, Expert i Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari escull l'opció de canviar la seva contrasenya	
	2. El sistema mostra un formulari amb dos camps, un per la nova contrasenya i l'altre per la seva confirmació.
3. L'usuari omple els dos camps.	
	4. El sistema comprova que les dades són correctes i emmagatzema la nova contrasenya.
Curs alternatiu	
4. El sistema detecta que els dos camps no són iguals i mostra un missatge d'error.	

Cas d'ús	Ajuda
Resum	Permet que un usuari accedeixi a l'ajuda
Actor	Administrador, Expert i Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari registrat de dins del sistema decideix accedir a l'ajuda.	
	2. El sistema mostra pàgina d'ajuda

Cas d'ús	About
Resum	Permet que un usuari consultar informació sobre el sistema
Actor	Administrador, Expert i Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari registrat de dins del sistema decideix accedir a l'about.	
	2. El sistema mostra pàgina d'informació sobre el sistema

Gestió d'usuaris

Cas d'ús	Crear usuari
Resum	Permet crear un usuari del sistema
Actor	Administrador

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un administrador escull l'opció de crear un nou usuari.	
	2. El sistema mostra un formulari amb les dades necessàries per crear el nou usuari, incloent dos camps per a la contrasenya i la seva reafirmació.
3. L'administrador escull els camps (nom d'usuari, contrasenya, dades personals i rols)	
	4. El sistema emmagatzema el nou usuari a la base de dades.
Curs alternatiu	
4. El sistema detecta que ja existeix un usuari amb aquell nom i mostra un missatge d'error.	
4. El sistema detecta que la contrasenya i la seva reafirmació no coincideixen i mostra un missatge d'error.	

Cas d'ús	Esborrar usuari
Resum	Permet esborrar un usuari del sistema
Actor	Administrador

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un administrador escull l'opció d'esborrar un usuari.	
	2. El sistema mostra una llista amb tots els usuaris existents.
3. L'administrador escull l'usuari a esborrar.	
	4. El sistema demana la confirmació a l'administrador sobre l'acció.
5. L'administrador reafirma l'opció d'esborrar.	
	6. El sistema comprova que existeixi l'usuari. Si es tracta d'un revisor, es desassignen les similituds que tenia assignades per a revisar. Finalment es mostra una llista amb tots els usuaris existents al sistema.
Curs alternatiu	
5. L'administrador decideix no esborrar l'usuari i s'acaba el cas d'ús.	
6. L'usuari escollit era l'únic administrador del sistema i no es permet esborrar.	

Cas d'ús	Editar usuari
Resum	Permet editar un usuari del sistema
Actor	Administrador

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un administrador escull l'opció d'editar un usuari.	
	2. El sistema mostra la informació de l'usuari escollit.
3. L'administrador modifica els camps de l'usuari.	
	4. El sistema comprova que existeixi l'usuari, emmagatzema la nova informació. Si era revisor i se li treu el rol, es desassignen totes les similituds que tenia per revisar. Finalment es mostra una llista amb tots els usuaris existents.
Curs alternatiu	
4. L'usuari escollit no existeix i el sistema mostra un missatge d'error.	
4. L'usuari escollit era l'únic administrador del sistema i no es permet esborrar.	

Cas d'ús	Canviar rols
Resum	Permet canviar els rols de tots els usuaris del sistema
Actor	Administrador

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un administrador escull l'opció de canviar els rols dels usuaris.	
	2. El sistema mostra una llista amb tots els usuaris i els rols que tenen.
3. L'administrador modifica els rols que convinguin i polsa l'opció de guardar els canvis.	
	4. El sistema comprova que tots els camps siguin vàlids i emmagatzema la informació a la base de dades. A més, si un revisor passa a no ser-ho se li desassignen les similituds que tenia per revisar.
Curs alternatiu	
4. El sistema comprova que s'ha deixat algun usuari sense cap rol assignat i es mostra un missatge d'error.	
4. El sistema comprova que no existeix cap usuari amb el rol d'administrador assignat i mostra un missatge d'error.	

Gestió de configuracions

Configuració de fonts de dades

Cas d'ús	Crear configuració de fonts de dades
Resum	Permet crear una nova configuració de fonts de dades
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari expert escull l'opció de crear una configuració de fonts de dades	
	2. El sistema dóna l'opció d'afegir tantes dades com es vulgui i afegeix un camp per a la descripció de la configuració.
3. L'usuari afegeix fonts de dades (csv) i defineix quin és el seu separador i si la primera fila conté els noms de les columnes i en veu una previsualització. Finalment omple el camp amb la descripció.	
	4. El sistema valida les dades, emmagatzema les fonts de dades i la configuració amb la descripció i un identificar únic a la base de dades i les lliga entre elles.
Curs alternatiu	
4. El sistema comprova que no s'ha afegit cap font de dades i mostra un missatge d'error.	

Cas d'ús	Esborrar configuració de fonts de dades
Resum	Permet esborrar una nova configuració de fonts de dades
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari expert escull l'opció d'esborrar una configuració de fonts de dades	
	2. El sistema llista les configuracions de fonts de dades que existeixen.
3. L'usuari escull la configuració que vol esborrar.	
	4. El sistema mostra un missatge d'advertència i demana confirmació.
5. L'usuari confirma la seva acció.	
	6. El sistema esborra la configuració de fonts de dades de la base de dades, també les fonts de dades i també esborra les configuracions de columnes de creuament, algorisme i exportació associades amb ella (mirar els casos d'ús corresponents).
Curs alternatiu	
5. L'usuari decideix no esborrar-la i s'acaba el cas d'ús.	

Configuració de columnes de creuament

Cas d'ús	Crear configuració de columnes de creuament
Resum	Permet crear una nova configuració de columnes de creuament
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari expert escull l'opció de crear una configuració de columnes de creuament.	
	2. El sistema mostra una llista amb totes les configuracions de fonts de dades existents.
3. L'usuari escull sobre quina configuració de fonts de dades vol crear la de columnes de creuament.	
	4. El sistema mostra un formulari amb les dades a omplir: columnes de creuament, descripció de la configuració. També mostra una previsualització de les fonts de dades de la configuració triada.
5. L'usuari tria les columnes de creuament assignant-lis columnes de les fonts de dades. A més, escull el tipus al que pertanyen, el comparador que es farà servir i el filtra que se li aplicarà a cada font. Per últim tria una descripció per la configuració.	
	6. El sistema comprova que almenys existeix una columna de creuament i mostra una llista amb els pesos per defecte que assignarà a cada columna.
7. L'usuari indica els pesos que vol aplicar.	
	8. El sistema emmagatzema les columnes de creuament i les lliga amb una nova configuració que té un identificador únic i la descripció triada. A més, aplica els filtres a totes les columnes i es guarda els valors filtrats.
Curs alternatiu	
6. El sistema comprova que no existeix cap columna de creuament i mostra un missatge d'error.	

Cas d'ús	Esborrar configuració de columnes de creuament
Resum	Permet esborrar una nova configuració de columnes de creuament
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari expert escull l'opció d'esborrar una configuració de columnes de creuament.	
	2. El sistema llista les configuracions de columnes de creuament que existeixen.
3. L'usuari escull la configuració que vol esborrar.	
	4. El sistema mostra un missatge d'advertència i demana confirmació.
5. L'usuari confirma la seva acció.	
	6. El sistema esborra la configuració de la base de dades, les columnes de creuament i els valors filtrats i també esborra les configuracions d'algorisme associades amb ella (mirar els cas d'ús corresponent).
Curs alternatiu	
5. L'usuari decideix no esborrar-la i s'acaba el cas d'ús.	

Configuració d'algorisme

Cas d'ús	Crear configuració d'algorisme
Resum	Permet crear una nova configuració d'algorisme
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari expert escull l'opció de crear una configuració d'algorisme.	
	2. El sistema mostra una llista amb totes les configuracions de columnes de creuament existents.
3. L'usuari escull sobre quina configuració de columnes de creuament vol crear la de columnes d'algorisme.	
	4. El sistema mostra un formulari amb la descripció de la configuració, el valor de la finestra i el pes mínim de similitud i llista les columnes de creuament que té la configuració de columnes de creuament triada.
5. L'usuari tria les columnes d'ordenació sobre les de creuament llistades, escull una descripció, un valor de la finestra i el pes mínim de similitud que vol obtenir en les similituds resultants.	
	6. El sistema comprova que almenys existeix una columna d'ordenació i emmagatzema la configuració amb un identificador únic i les dades introduïdes. A més, emmagatzema les columnes d'ordenació i les associa amb la configuració.
Curs alternatiu	
6. El sistema comprova que no existeix cap columna d'ordenació i mostra un missatge d'error.	
6. El sistema comprova que el pes mínim no està entre 0 i 100 i mostra un missatge d'error.	
6. El sistema comprova que la finestra té un valor menor o igual a 0 i mostra un missatge d'error.	

Cas d'ús	Esborrar configuració d'algorisme
Resum	Permet esborrar una nova configuració d'algorisme
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari expert escull l'opció d'esborrar una configuració d'algorisme.	
	2. El sistema llista les configuracions de columnes d'algorisme que existeixen.
3. L'usuari escull la configuració que vol esborrar.	
	4. El sistema mostra un missatge d'advertència i demana confirmació.
5. L'usuari confirma la seva acció.	
	6. El sistema esborra la configuració de la base de dades, les columnes d'ordenació i totes les similituds resultants d'haver executat la configuració existents.
Curs alternatiu	
5. L'usuari decideix no esborrar-la i s'acaba el cas d'ús.	

Execució

Cas d'ús	Execució
Resum	Permet executar un procés de Record Linkage
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un usuari escull l'opció d'executar un procés de Record Linkage.	
	2. El sistema mostra una llista amb les configuracions d'algorisme existents.
3. L'usuari escull sobre quina configuració d'algorisme vol fer l'execució.	
	4. El sistema executa el Record Linkage amb els paràmetres de la configuració triada, la qual està relacionada amb una de columnes de creuament i una de fonts de dades. Finalment mostra informació sobre l'execució (temps, nombre de similituds, de grups,...)

Revisió

Cas d'ús	Assignar similituds
Resum	Permet assignar les similituds als revisors
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció d'assignar les similituds d'una execució a revisors.	
	2. El sistema llista les configuracions d'algorisme existents que han estat executades.
3. L'expert escull sobre quina vol assignar revisors.	
	4. El sistema mostra una llista amb tots els revisors existents.
5. L'expert tria a quins revisors se'ls hi vol assignar similituds.	
	6. El sistema reparteix els grups de similituds existents entre els revisors triats de forma equitativa en número de grups, emmagatzema les assignacions i les hi mostra a l'usuari.
7. L'expert veu les assignacions i canvia les que interessen podent fins i tot no assignar alguns grups a ningú.	
	8. El sistema emmagatzema els canvis fets per l'usuari.

Cas d'ús	Revisar per similituds
Resum	Permet revisar els resultats d'una execució ordenada per similituds
Actor	Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un revisor escull l'opció de revisar.	
	2. El sistema llista configuracions d'algorisme executades que tenen com a resultat alguna similitud assignada a aquell revisor.
3. El revisor escull quina vol revisar.	
	4. El sistema mostra una llista amb totes les similituds assignades al revisor de la configuració que ha escollit ordenades descendentment per grau de similitud i les posa totes amb estat *indecís*.
5. El revisor canvia els estats de les similituds pel que desitgi entre: *acceptat*, *rebutjat* o *indecís*.	
	6. El sistema detecta cada canvi d'estat de les similituds i els emmagatzema a la base de dades.

Cas d'ús	Revisar per grups
Resum	Permet revisar els resultats d'una execució ordenada per grups de similituds
Actor	Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un revisor escull l'opció de revisar.	
	2. El sistema llista configuracions d'algorisme executades que tenen com a resultat alguna similitud assignada a aquell revisor.
3. El revisor escull quina vol revisar.	
	4. El sistema mostra una llista amb totes les similituds assignades al revisor de la configuració que ha escollit ordenades pel número de grup al que pertanyen i, dins d'ell, descendentment per grau de similitud i les posa totes amb estat *indecís*.
5. El revisor canvia els estats de les similituds pel que desitgi entre: *acceptat*, *rebutjat* o *indecís*.	
	6. El sistema detecta cada canvi d'estat de les similituds i els emmagatzema a la base de dades.

Exportació

Cas d'ús	Crear configuració d'exportació
Resum	Permet crear una nova configuració d'exportació
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció de crear una configuració d'exportació.	
	2. El sistema mostra una llista amb totes les configuracions de fonts de dades existents.
3. L'expert escull sobre quina configuració de fonts de dades vol crear la d'exportació.	
	4. El sistema mostra un formulari amb les dades a omplir: columnes d'exportació, descripció de la configuració i altres camps (mostrar pes i mostrar estat). També mostra una previsualització de les fonts de dades de la configuració triada.
5. L'expert tria les columnes d'exportació assignant-lis columnes de les fonts de dades i escollint si són editables o no i omple els altres camps.	
	6. El sistema comprova que almenys existeix una columna d'exportació i emmagatzema la nova configuració assignant-li un identificador únic lligant-la amb la de fonts de dades triada.
Curs alternatiu	
6. El sistema comprova que no existeix cap columna d'exportació i mostra un missatge d'error.	

Cas d'ús	Esborrar configuració d'exportació
Resum	Permet esborrar una configuració d'exportació existent
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció d'esborrar una configuració d'exportació.	
	2. El sistema mostra una llista amb totes les configuracions d'exportació existents.
3. L'expert escull la configuració a esborrar.	
	4. El sistema demana la confirmació a l'administrador sobre l'acció.
5. L'expert reafirma l'opció d'esborrar.	
	6. El sistema esborrar la configuració juntament amb els valors d'exportació que tenia associats.
Curs alternatiu	
5. L'expert decideix no esborrar i s'acaba el cas d'ús.	

Cas d'ús	Escollir valors
Resum	Permet escollir els valors que s'exportaran per a cada similitud trobada
Actor	Revisor

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un revisor escull l'opció d'exportar.	
	2. El sistema mostra una llista amb les configuracions d'algorisme existents a les quals té alguna similitud associada el revisor.
3. El revisor escull la configuració sobre la qual vol exportar els seus resultats.	
	4. El sistema mostra una llista amb les configuracions d'exportació existents.
5. El revisor escull la configuració d'exportació que vol aplicar.	
	6. El sistema mostra una llista amb els valors d'exportació possibles per a cada similitud segons la configuració d'exportació triada i només d'aquelles que el revisor té assignades. Si no s'han escollit valors per a cada similitud encara, en crea els per defecte.
7. El revisor canviar els valors de les columnes d'exportació que desitgi per a cada similitud.	
	8. El sistema detecta cada canvi de valor de les columnes d'exportació de les similituds i els emmagatzema a la base de dades.

Cas d'ús	Exportar fitxer
Resum	Permet exportar un fitxer de resultats
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció d'exportar.	
	2. El sistema mostra una llista amb les configuracions d'algorisme existents.
3. L'expert escull la configuració sobre la qual vol exportar els seus resultats.	
	4. El sistema mostra una llista amb les configuracions d'exportació existents.
5. L'expert escull la configuració d'exportació que vol aplicar.	
	6. El sistema pregunta per l'estat de les similituds que es volen exportar.
7. L'expert escull els estats que vol exportar.	
	8. El sistema mostra una llista amb els valors d'exportació escollits per a cada similitud que té com a estat un dels triats i dona l'opció d'exportar a fitxer. Si no s'han escollit valors per a cada similitud, els crea amb els per defecte.
9. L'expert escull el format en que vol exportar el fitxer.	
	10. El sistema mostra un diàleg de descàrrega del fitxer.
Curs alternatiu	
8. El sistema comprova que no s'ha escollit cap estat i mostra un missatge d'error.	

Gestió de filtres

Cas d'ús	Crear filtre
Resum	Permet crear un nou filtre de dades
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció de crear un filtre.	
	2. El sistema mostra un formulari amb les dades necessàries del filtre: nom, descripció, tipus de columnes de creuament a les que pot aplicar-se. A més, mostra un editor de codi i dóna la opció de provar el codi introduït.
3. L'expert introdueix les dades requerides. També prova el codi del filtre introduït per a comprovar la correctesa.	
	4. El sistema comprova que no existeixi cap filtre amb el mateix nom i l'emmagatzema a base de dades.
Curs alternatiu	
3. L'expert no està satisfet amb el codi i torna a editar-lo i segueix en el pas 3. 4. El sistema detecta que ja existeix un filtre amb el nom escollit i mostra un missatge d'error.	

Cas d'ús	Editar filtre
Resum	Permet editar un filtre de dades
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció d'editar un filtre.	
	2. El sistema mostra un formulari amb les dades del filtre: nom, descripció, tipus de columnes de creuament a les que pot aplicar-se. A més, mostra un editor amb el codi i dóna la opció de provar-lo.
3. L'expert editar les dades desitjades. També prova el codi del filtre introduït per a comprovar la correctesa.	
	4. El sistema emmagatzema els canvis a la base de dades. Si el nom del filtre és el mateix que el que es volia editar, actualitza les seves dades amb les escollides per l'expert; si el nom és diferent es crea un nou filtre amb aquell nom i les dades escollides i es manté el filtre que es volia editar.
Curs alternatiu	
3. L'expert no està satisfet amb el codi i torna a editar-lo i segueix en el pas 3. 4. El sistema detecta que ja existeix un filtre amb el nom escollit i que aquest no és el que es volia editar i mostra un missatge d'error.	

Cas d'ús	Visualitzar filtre
Resum	Permet visualitzar un filtre de dades i provar-lo
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció de visualitzar un filtre.	
	2. El sistema mostra les dades del filtre: nom, descripció, tipus de columnes de creuament a les que pot aplicar-se. A més, mostra el seu codi i dóna la opció de provar-lo.

Cas d'ús	Esborrar filtre
Resum	Permet esborrar un filtre de dades
Actor	Expert

Curs típic d'esdeveniments	
Accions de l'actor	Accions del sistema
1. Un expert escull l'opció d'esborrar un filtre.	
	2. El sistema llista tots els filtres existents.
3. L'expert escull el filtre que vol esborrar.	
	4. El sistema esborra el filtre de base de dades.
Curs alternatiu	
4. El sistema detecta que el filtre està sent utilitzat per alguna columna de creuament i mostra un missatge d'error.	

4.4 Model conceptual

El model conceptual d'un sistema software permet dibuixar quina serà l'estructura de les classes i com es relacionen entre elles, a més dels atributs principals que les conformaran. Les operacions d'aquestes classes no les presentarem, degut a que la majoria d'elles només són per a posar i obtenir els valors dels seus atributs. Això es degut a que es dissenyarà tota l'aplicació en diverses capes, tal i com s'explica al Capítol 5, de manera que el model conceptual quedarà separat de la funcionalitat.

És habitual que el model conceptual sigui d'una mida considerable atesa l'envergadura dels projectes que se solen construir, en els quals hi intervé molta informació diferent i s'ofereixen diverses funcionalitats als usuaris finals. És per això que es presentarà el model conceptual dividit en diferents mòduls. Cal tenir en compte que, cada mòdul, contindrà un conjunt de classes que comparteixen un elevat nombre de dades i, alhora, estan destinades a constituir funcionalitats de forma conjunta. Així, podem observar que per a diferents mòduls existiran classes repetides, degut a que aquestes intervenen en tots ells.

Fonts de dades Aquest primer mòdul representa l'estructura que es tindrà per a definir les fonts de dades i les configuracions a les que pertanyen. A més, com es pot observar, ja s'ha tingut en compte una possible extensió del projecte que s'està definint, al existir diverses formats de fonts possibles : CSV i base de dades. Tot i així, recordem que en aquesta primera versió només estarà disponible el primer dels formats. Així, la Figura 4.6 mostra el model que correspon a aquest mòdul.

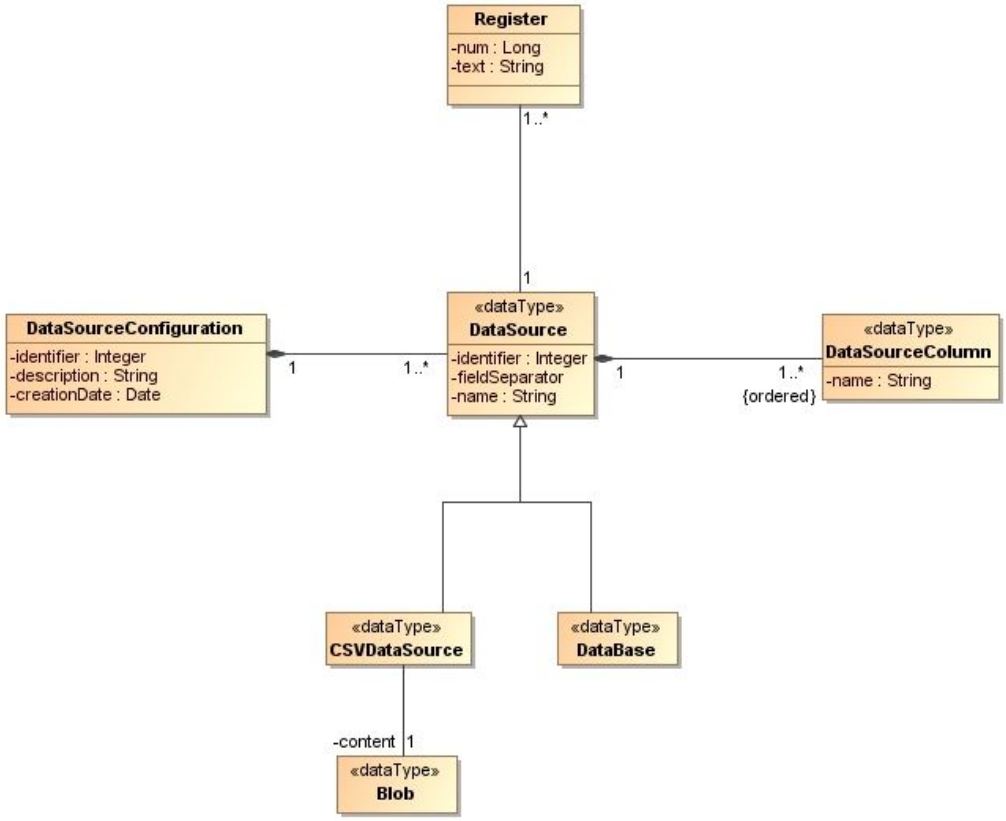


Figura 4.6: Model conceptual estructural de les fonts de dades

Creuament En aquest segon mòdul s'il·lustrarà la part del model conceptual que està relacionada amb la configuració de columnes de creuament. Recordem que, per crear aquest tipus de columnes, s'ha de fer a partir de les columnes que posseeixen les fonts de dades triades i que, per tant, existeix un lligam entre la configuració de creuament i la de fonts de dades. A més, al definir les columnes de creuament, es procedeix a fer el **preprocés** de dades, el qual implica haver d'emmagatzemar aquests valors filtrats per a cada registre en una entitat que anomenem *CrossColumnField*. Així, la Figura 4.7 mostra com s'estructurarà aquesta informació.

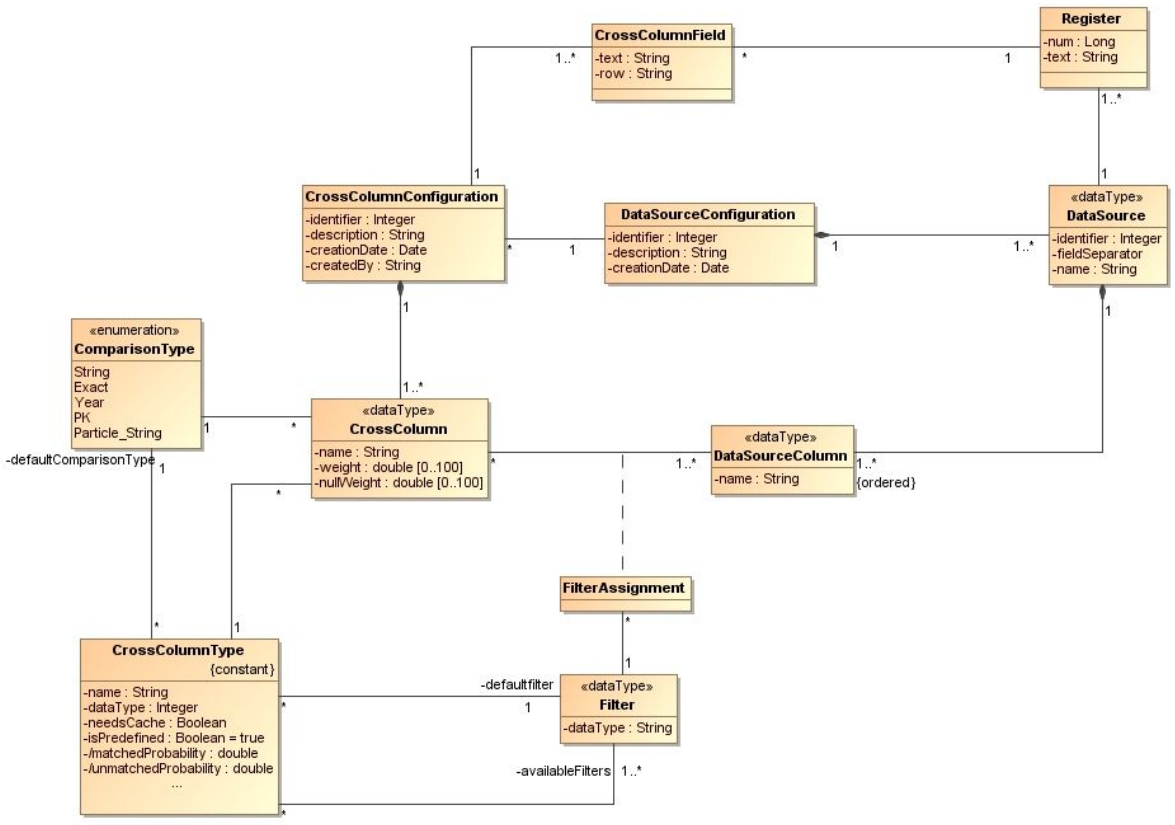


Figura 4.7: Model conceptual estructural de la configuració de columnes de creuament

Algorisme Aquest tercer mòdul representa la creació d'una configuració d'algorisme, la qual només pot existir si abans se n'ha creat una de columnes de creuament, tal i com s'explica al capítol 3. Per a donar la opció en un futur de l'existència de diversos algorismes a la hora de fer les comparacions de registres, s'ha definit una entitat comuna de la qual hereten les diferents implementacions d'algorisme i de la qual només existeix una: la de *Sliding Window*. A la Figura 4.8 veiem el que s'acaba d'explicar de forma gràfica.

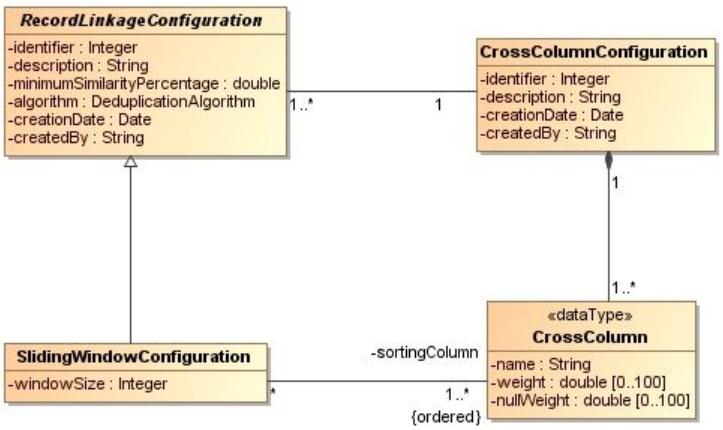


Figura 4.8: Model conceptual estructural d'una configuració d'algorisme

Revisió Aquest quart mòdul engloba tot el model necessari per a fer la **revisió**. A més, es mostrarà també com es representen els usuaris del sistema, el quals tindran diversos rols, ja que a ells se'ls hi haurà d'assignar un rang de grups de similituds a revisar. La Figura 4.9 il·lustra aquesta representació del model.

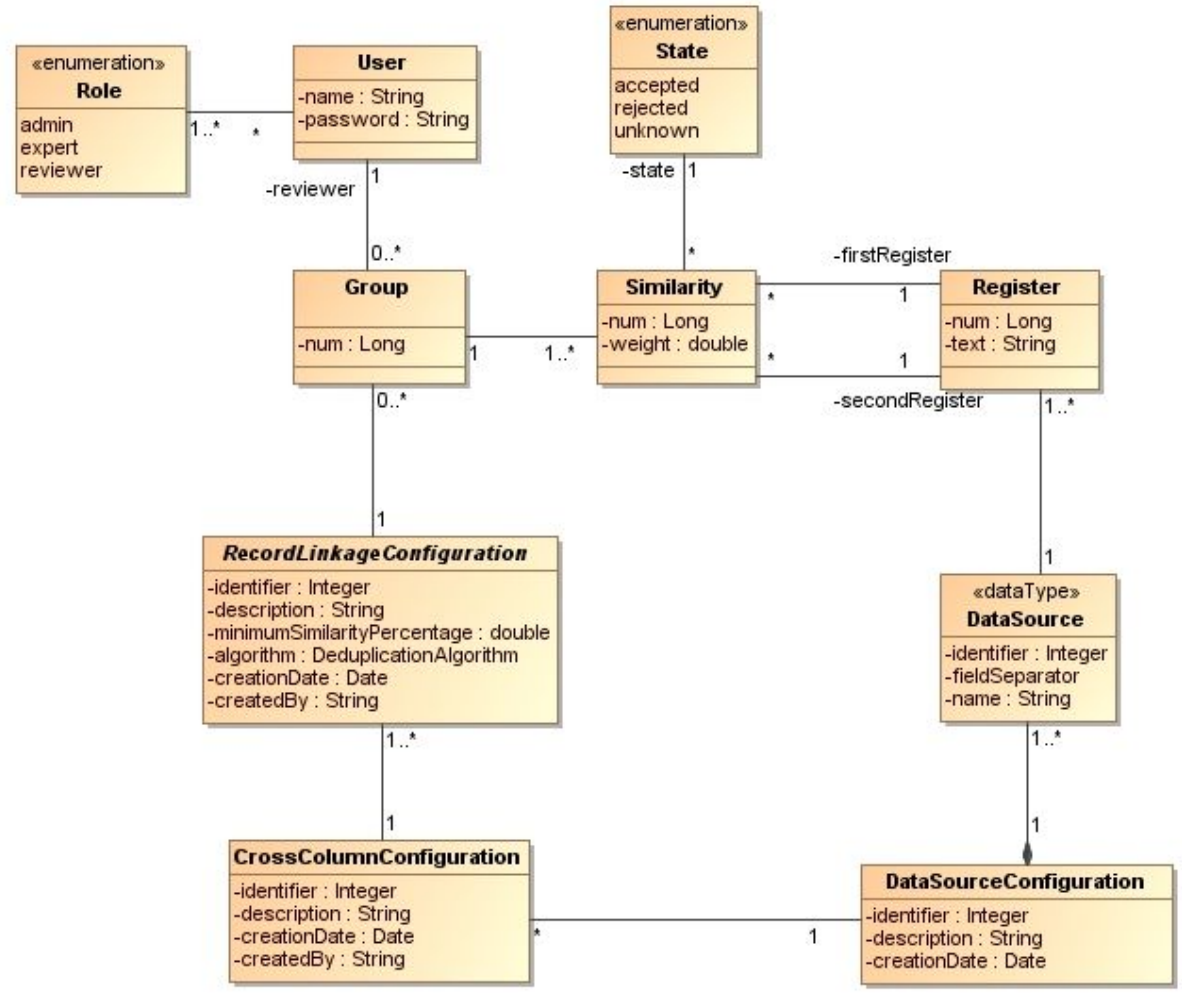


Figura 4.9: Model conceptual estructural de la revisió de similituds

Exportació El cinquè i darrer mòdul serveix per a representar l'**exportació** de les similituds trobades. Per tant, han d'existir totes les entitats referents a la creació de la seva configuració i també les necessàries per a escollir i emmagatzemar els valors que es volen exportar per a cada similitud. A la Figura 4.10 s'observa la representació de totes les entitats relaciones amb l'exportació.

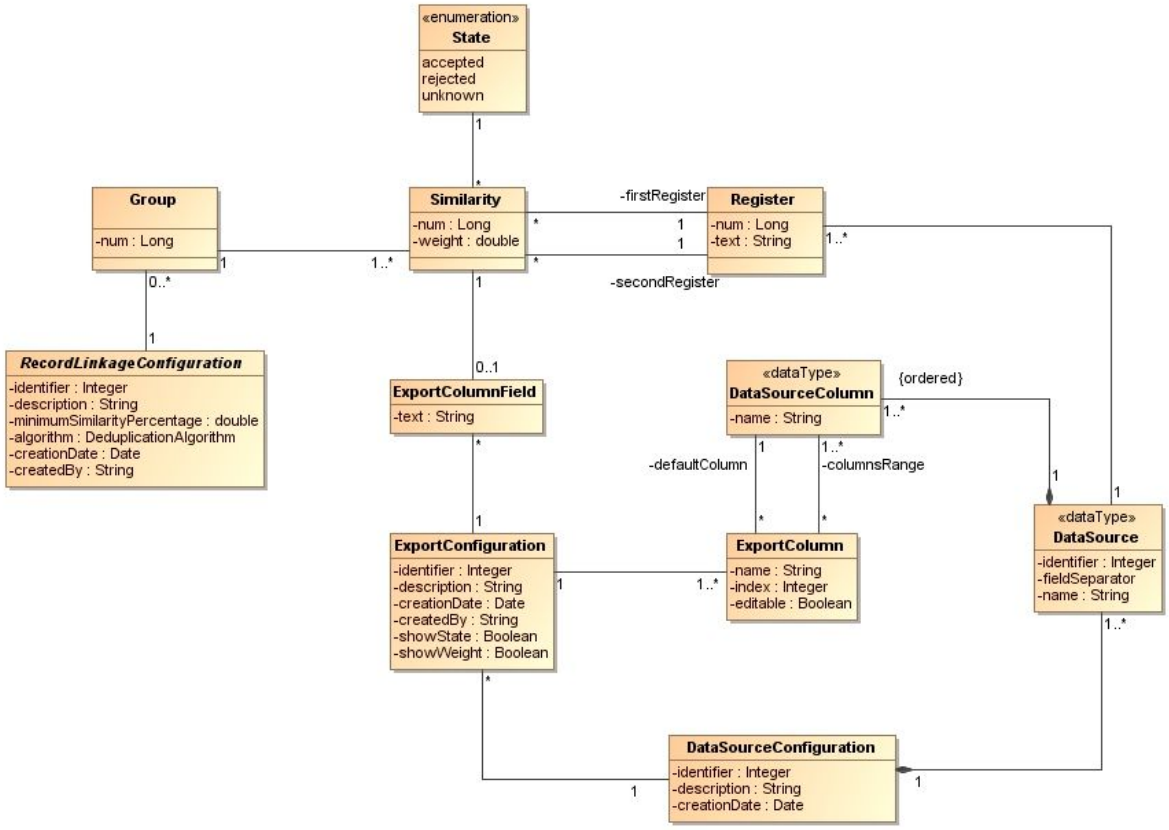


Figura 4.10: Model conceptual estructural de l'exportació

Model complet En els cinc mòduls anteriors s'han vist representats diferents parts del model. D'aquesta manera, s'ha permés visualitzar i entendre millor els elements que hi participen en cadascun d'ells. Tot i així, a vegades ha aparescut una mateixa classe a més d'un mòdul, degut a que jugava un paper important a diversos d'ells. A la Figura 4.11 (de costat) podem observar el model conceptual complet, en el qual no apareixeran classes repetides i es podrà fer una visió global de tota l'aplicació.

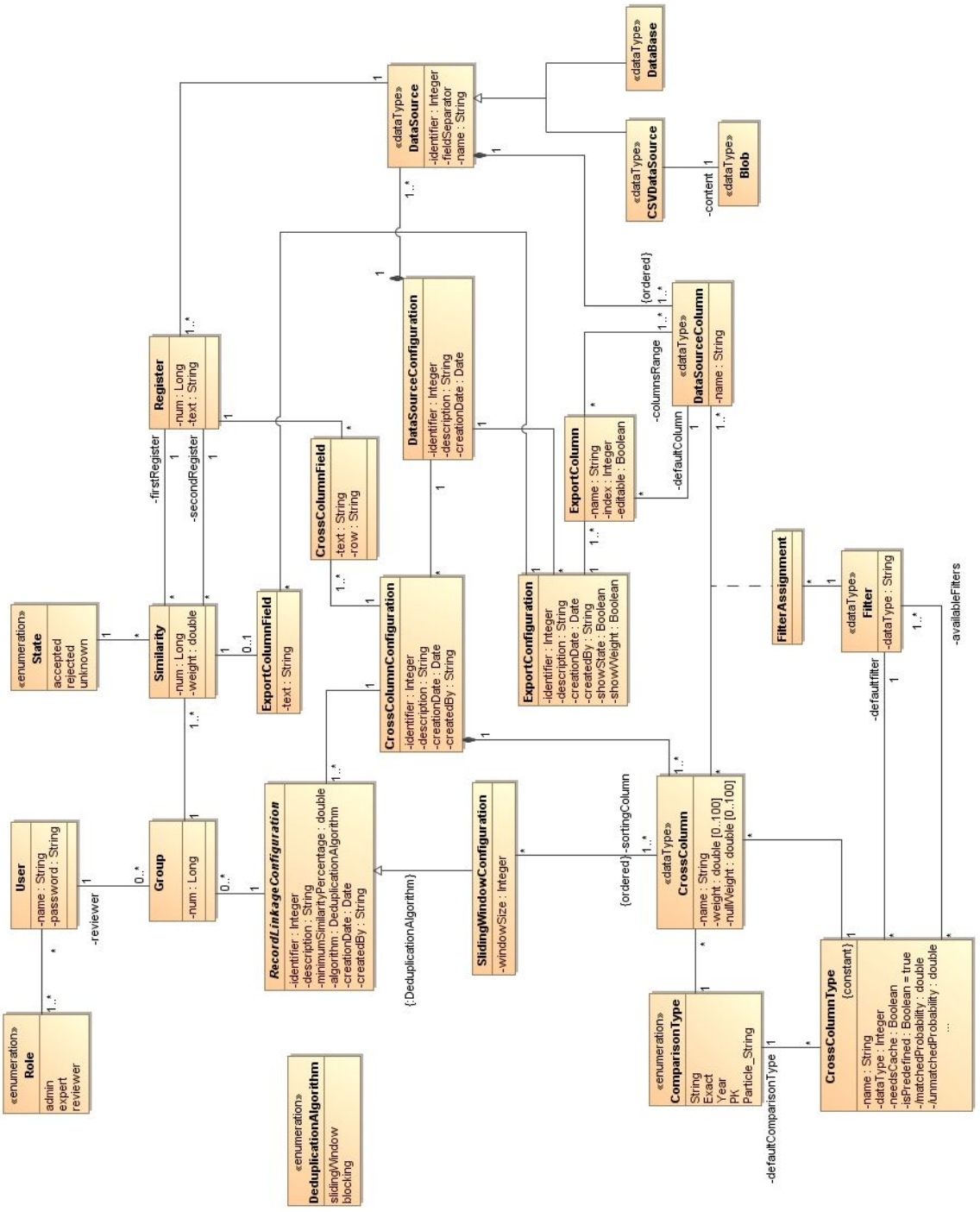


Figura 4.11: Model conceptual estructural de Daurum 5.0

Capítol 5

Disseny

Una vegada finalitzada l'especificació de l'aplicació, en la que s'ha determinat què és el que farà, és moment de descriure com ho farà per a aconseguir els objectius i això s'engloba dins del disseny. Així, la fase de disseny és la que assegura una futura implementació correcta i ràpida.

5.1 Introducció

En primer lloc, cal comentar que el llenguatge de programació que es farà servir per a dur a terme aquest projecte és el de Java [9], com ja s'ha esmentat durant l'especificació.

L'utilització d'aquest llenguatge està molt extesa en l'àmbit de les aplicacions web on el sistema està instal·lat a un servidor, degut a la seva rapidesa comparada amb la d'altres opcions com poden ser els components *CGI* i a que els servlets, explicats més endavant, s'executen dins del mateix procés de la *JVM* (*Java Virtual Machine*) millorant així notablement el rendiment i reduïnt la càrrega computacional i de memòria requerides. A més, gràcies a la seva expansió, s'han construït molts frameworks, com Struts2 [4] el qual serà útil per aquest projecte, molt elaborats que permeten aconseguir un entorn de treball molt més poderós i una segmentació de les diferents especialitzacions necessàries per crear una nova aplicació (desenvolupadors, dissenyadors gràfics, ...) i també ajuden en la reutilització i robustesa del codi generat.

Per últim, comentar també que un altre punt a favor de Java és la independència que té respecte a la plataforma on s'implanta el nou sistema, ja que aquest llenguatge al ser compilat genera un codi anomenat *bytecode* que serà interpretat i executat per una màquina virtual (*JVM*), que està escrita en el codi natiu de la plataforma destí i, per tant, permet ser executat en qualsevol d'elles que disposi una màquina virutal.

5.2 Patró Model Vista Controlador

El Model Vista Controlador (MVC) és un patró de disseny del software que separa l'arquitectura d'un sistema en tres components diferents: el Model, la Vista i el Controlador.

El **Model** és la lògica de negoci, que sovint involucra també l'accés a l'emmagatzament de dades en un medi persistent com poden ser les bases de dades. Així doncs, aquest primer component encapsula les dades i les funcionalitats del sistema. Dins d'ell, es poden diferenciar també diferents entitats: les pròpies del model conceptual (*vegeu Capítol 4.4.1*), les relacionades amb l'emmagatzamament d'informació i les que aporten informació a la Vista.

La **Vista** s'encarrega de presentar la interfície d'usuari en un format adequat per a interactuar. En sistemes web típicament es tracta de *HTML*, però poden existir altres tipus de vistes. Normalment només es poden fer operacions bàsiques com són sentències condicionals, iteratives o de format.

El **Controlador** és el que determina el fluxe general de l'aplicació. Té la tasca de respondre als events, usualment accions fetes per l'usuari des de la Vista, i invoca canvis al Model i possiblement també a la Vista.

Així, el fluxe habitual que segueix una aplicació que està estructurada seguint l'arquitectura del patró MVC és el següent:

- L'usuari interactua amb la interfície (Vista) i llença algún event (polsant una tecla, pitjant el ratolí,...).
- El Controlador rep aquesta acció produïda per l'usuari mitjançant un gestor d'events.
- El Controlador actualitza el Model segons l'acció executada per l'usuari. És freqüent que els controladors complexes estiguin estructurats utilitzant el patró *Command Pattern*, el qual permet solicitar una operació d'un objecte sense conèixer realment el seu contingut ni el receptor d'ella.
- El Controlador delega a la Vista la tasca de desplegar la interfície d'usuari, la qual obté les dades a mostrar del Model.
- La interfície d'usuari espera noves interaccions, començant de nou aquest cicle.

Aquest patró presentat és utilitzat freqüentment en sistemes web JEE (*Java Enterprise Edition*) i té beneficis diversos entre els que destaquen:

- La distribució de l'esforç de desenvolupament fins a cert punt, tal que els canvis d'implementació en una part de l'aplicació web no requereixin canvis en altres parts. Així, aporta independència entre el Model, la Vista i el Controlador.
- Facilitat en la migració de antigues aplicacions, ja que la Vista està separada del Model i del Controlador i pot ser més tolerable a diferents plataformas i categories d'usuaris.
- El disseny MVC té una estructura organitzada que millora l'escalabilitat (construir aplicacions més grans) i és fàcil de modificar i mantenir debut a la clara separació de tasques.

Struts 2

Struts 2 [22] és un *framework* de suport per al desenvolupament d'aplicacions web sota el patró MVC descrit anteriorment i el qual s'ha decidit utilitzar en aquest projecte degut a la seva gran solidesa i a la simplificació que fa per a la contrucció d'un sistema web com el que es vol per a aquest PFC. A més, s'integra amb Ajax i, per tant, permet tenir aplicacions més interactives i amigables pels usuaris finals.

Primer de tot, cal introduir un parell de conceptes per a entendre el funcionament d'aquest framework: els *Servlets* i les pàgines *JSP*.

Els *Servlets* són una tecnologia que permet mapejar una URL en una classe especial en la qual els seus mètodes són cridats i que s'utilitzen per a generar pàgines web de forma dinàmica. Això implica generar codi HTML dins d'aquestes classes especials, fet que implica fer un gran manteniment i resulta massa costós per a utilitzar-se.

Per això, van sorgir les pàgines JSP, *Java Server Page*, [1] que permeten fer el contrari: introduir codi Java dins del codi HTML. A més, per a permetre la reutilització de codi i una millor estructuració, es van introduir nous tags (etiquetes) que permetien accedir a codi Java des de les pàgines JSP. El funcionamient general de la tecnologia JSP és que el servidor d'aplicacions, com pot ser *Tomcat,* interpreta el codi contingut a la pàgina i construeix el codi Java del servlet, que generarà el codi HTML que es mostrarà al navegador dels usuaris i aquest només serà generat una vegada: la primera vegada que es consulti la pàgina en qüestio.

El framework de Struts 2 està basat en accions (*action-based*). Això vol dir que per a cada petició que fan els usuaris del sistema al accedir a una URL existeix un mapeig entre aquella URL i una unitat de treball, la qual

anomenem *action* i que forma part del **Model**. Els *Servlets* formen part del **Controlador**, que és l'encarregat de controlar de forma centralitzada totes les peticions i redirigir-les cap a un flux concret segons l'acció que desitgi fer l'usuari. La tasca que farà una *action* serà la d'executar una certa funcionalitat que es vol donar quan s'accedeix a una URL concreta, accedint a la sessió i la petició *HTTP* i recollint la informació establerta pels usuaris. A més, també modificarà les classes que corresponen amb la lògica de negoci i les farà persistents a la base de dades si és necessari. Finalment, la *action* retornarà un resultat que serà un simple String que el **Controlador** recollirà i, segons el seu valor, farà que a la **Vista** es mostri una JSP o una altra. Per últim, aquesta JSP accedirà al **Model** per a mostrar la informació que necessiti a l'usuari.

La Figura 5.1 mostra el procés que es du a terme en una aplicació que utilitza Struts 2. Com es pot observar, després de que un usuari faci una petició accedint a una URL de l'aplicació des del seu ordinador, aquesta es dirigeix cap al Controlador. Aquest té definit un mapeig URL-*action* amb la qual cosa es podrà saber quina acció s'ha de dur a terme. Abans d'accedir a la *action* s'executaran una sèrie d'interceptors que també estan definits en aquest mapeig i que serveixen per a fer un preprocés i un postprocés sobre la funcionalitat que aporta la *action* en qüestió (al Capítol 5.3 s'explica més extensament com funcionen els interceptors, per a què serveixen i quins es faran servir). Així, una vegada executats aquests, s'accedeix a la *action* omplint-la amb els paràmetres entrats per l'usuari (*setXxx()*) i s'executa mitjançant el mètode *execute()*, ajudant-nos de *Command Pattern* tal i com s'ha explicat amb anterioritat. L'execució d'una *action* pot implicar una modificació de les dades i s'acaba retornant un String que servirà per a determinar quina JSP s'ha de mostrar a l'interfície d'usuari. Aquesta JSP accedirà a la *action* per a recollir les dades provinents del Model que es visualitzaran (*getXxx()*).

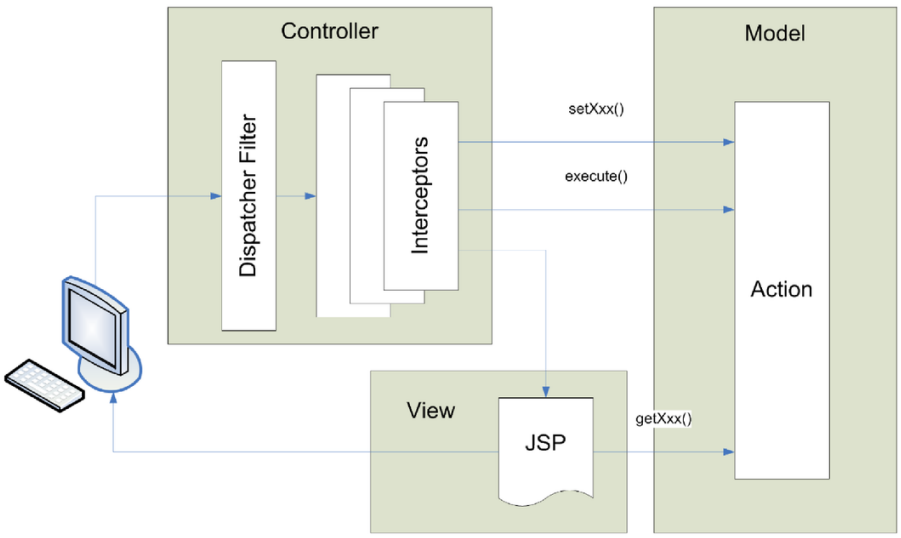


Figura 5.1: Diagrama de l'execució d'una acció utilitzant Struts2

5.3 Model

La capa del Model està composada per classes Java, les quals poden venir directament des del model conceptual (en endavant classes de domini) que s'ha definit anteriorment o poden haver sorgit per a proporcionar una certa funcionalitat al sistema. Així, per a cada operació del sistema, tindrem una classe, la qual anomenem *action*, que permet treballar amb les classes de domini i accedir a la base de dades per emmagatzemar els canvis que es realitzin. Per tant, aquestes *actions* el que representen són funcionalitats del sistema. A més, també existirà un tercer tipus de classe, el qual anomenem *preparer*. Aquesta classe ens serà útil per a la capa de la Vista, ja que serveix per a carregar informació provinent del Model per a mostrar-la mitjançant una JSP.

Així doncs, la part més important del sistema que s'està construïnt, és la del Model, ja que serà aquesta la que proporcionarà noves funcionalitats i treballarà amb la informació per a fer-la persistent i poder-la obtenir per a mostrar-la als usuaris finals.

5.3.1 Diagrames de seqüència

Les descripcions realitzades en el model de casos d'ús són un exemple clar de definicions amb l'objectiu de realitzar les últimes tasques del cicle de vida de la construcció d'un software. Per cada cas d'ús rellevant és necessari definir el diagrama de seqüència de totes les operacions i dades que hi intervenen. Per tal de simplicar aquest model, tan sols es representaran els diagrames de seqüència d'aquells casos d'ús que realment sigui interessant observar-ne el detall. S'han triat tres casos dels qual se n'exposaran els seus diagrames, relatius a diferents parts del procés de Record Linkage i que, per tant, és d'interés saber-ne el seu disseny.

Finalment, cal afegir que els diagrames de seqüència no representen un disseny exhaustiu sinó que són una visió molt aproximada del que serà el resultat final de la implementació. Amb l'objectiu de simplificar els diagrames, tot el que té a veure amb la base de dades ha quedat agrupat en una classe anomenada *BD* i s'ha obviat el que fa referència amb la part visual de l'aplicació anomenant-la només amb crides a funcions com *mostraPantalla()*, degut a que la part visual serà codi HTML. Per a més informació sobre aquesta part es pot consultar el Capítol 5.4, on es mostra amb exemples com funciona la generació de codi per a les vistes.

Com s'ha explicat amb anterioritat, són les *actions* les que executen una funcionalitat i, per a fer-ho, s'ha d'invocar el seu mètode *execute*(). Veurem en els diagrames doncs, aquesta invocació, amb la qual podrem identificar les *actions* existents. Per a posar els valors dels seus atributs, s'ha fet representant que el Controlador és l'encarregat, tot i que en la pràctica existirà un interceptor que farà aquesta tasca. En el Capítol 5.5 s'expliquen en detall els interceptors.

El primer diagrama de seqüència és referent a la creació de les configuracions necessàries i mostrarà de forma simplificada com es podrà crear una configuració de columnes de creuament. Se suposarà que ja s'han escollit les columnes que es volen afegir i, per tant, només es mostrarà l'últim pas que és el de emmagatzemar tota la informació a base de dades. Així, la Figura 5.2 mostra el model de comportament descrit.

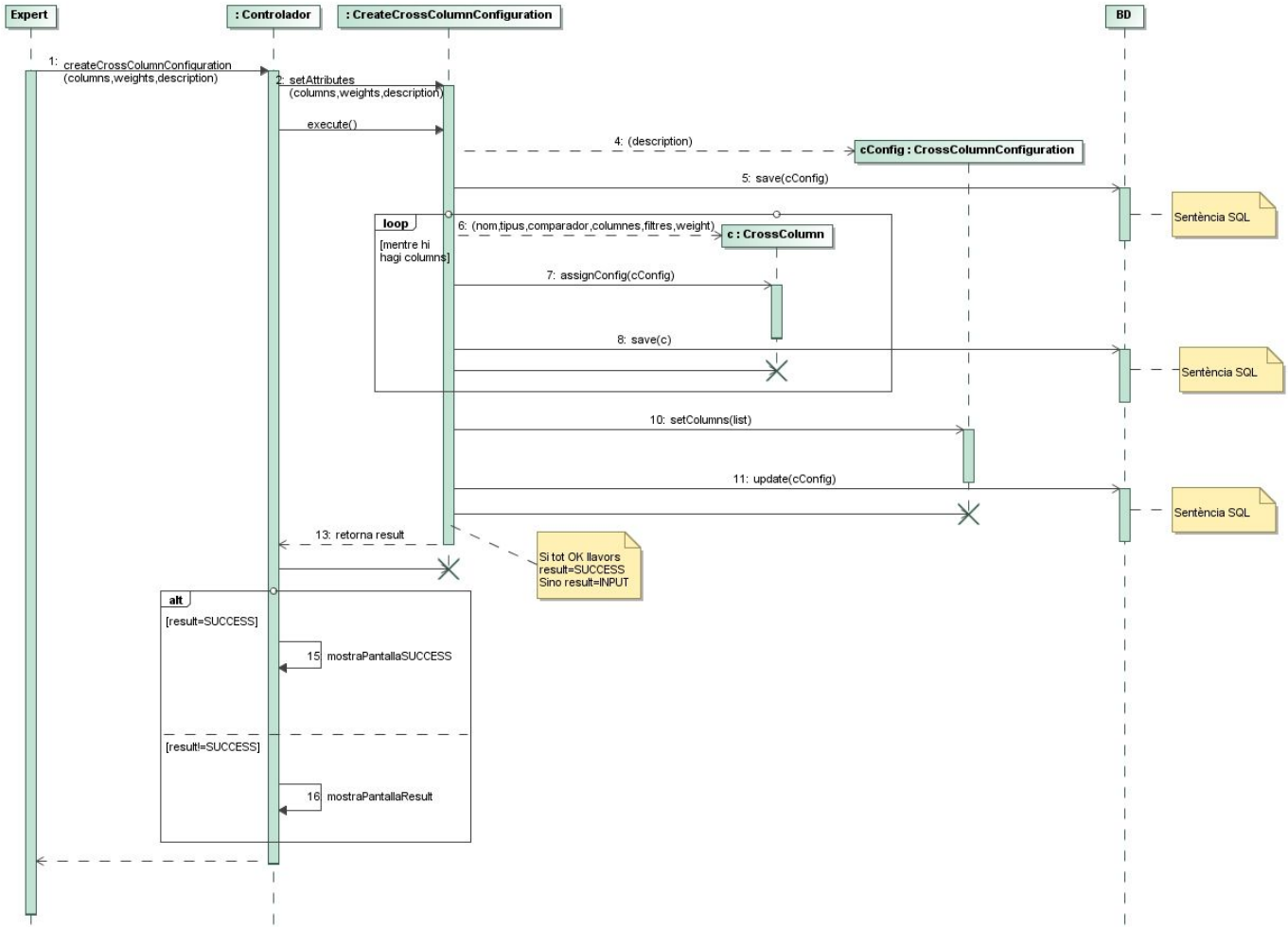


Figura 5.2: Diagrama de seqüència de la creació d'una configuració de columnes de creuament.

Com es pot observar, el comportament compleix el descrit amb anterioritat en el Capítol 5.2, ja que és el Controlador qui decideix en funció del resultat de retorn de *CreateCrossColumnConfiguration*, que és una *action*, quina pantalla es motrarà a l'usuari. A més, veiem que qui comença aquest procés és un usuari **expert** que és l'únic que té el permís per a fer-ho. Cal destacar també que les sentències SQL generades per no s'han mostrat ja que generalment serà un framework qui ho farà per nosaltres (més informació al Capítol 5.3.2).

El segon diagrama de seqüència que s'ha decidit mostrar és el de l'execució de Record Linkage. Aquí s'ha utilitzat el polimorfisme que facilita Java per a poder afegir a l'execució els paràmetres que corresponen a cada mètode de comparació de registres. Recordem que en aquest projecte només s'utilitzarà el mètode de Sliding Window, el qual té com a paràmetres la mida de la finestra i les columnes d'ordenació. Tot i així, tal i com ja s'ha comentat amb anterioritat, el sistema està preparat per acceptar futurs nous mètodes de comparació. En el segon pas, la funció *executaPas2()* trobem present el poliformisme en forma de patró Plantilla. Així doncs, la Figura 5.3 mostra el model de comportament de l'execució i les següents figures mostren en detall funcions existents en aquesta primera.

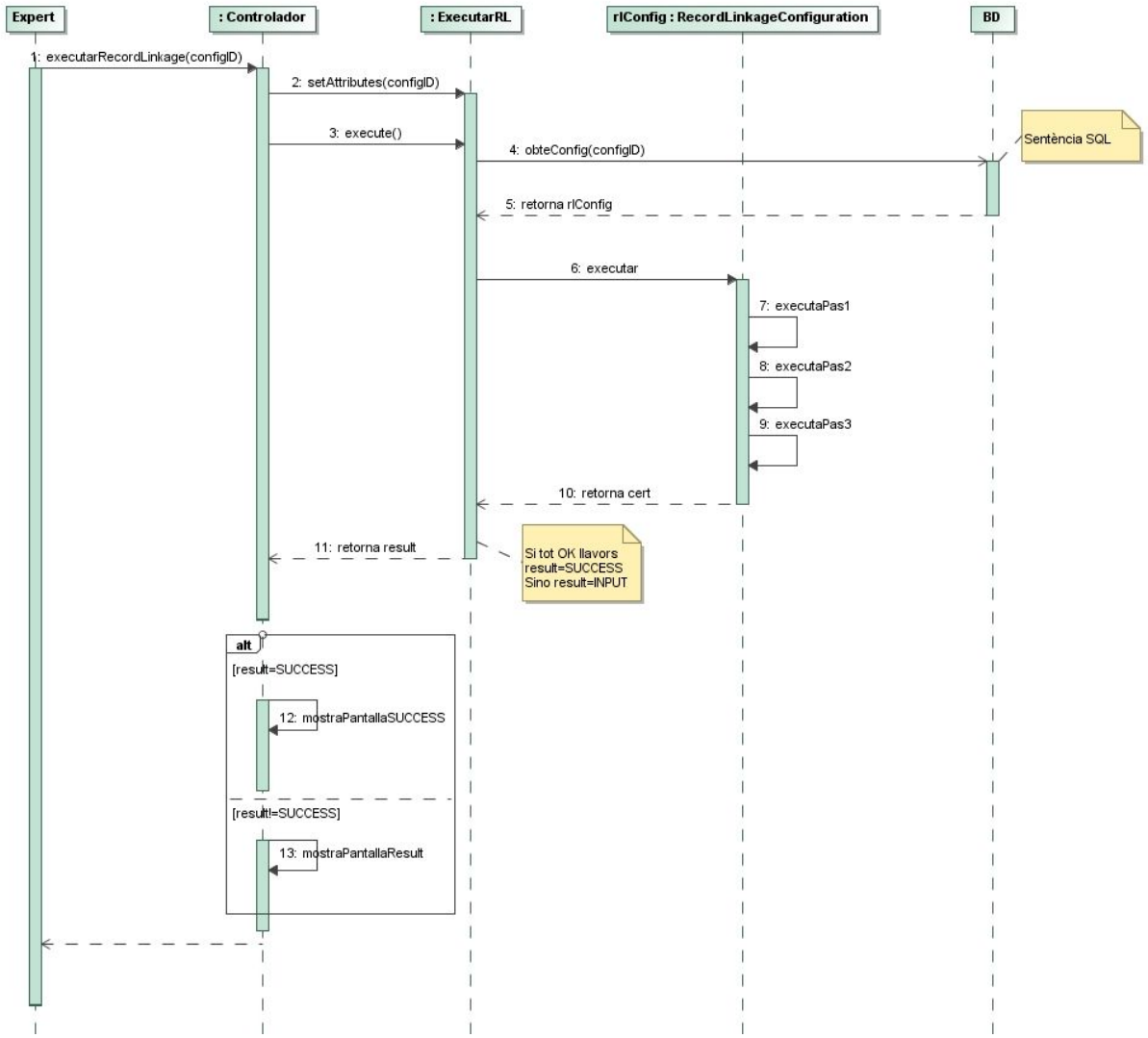


Figura 5.3: Diagrama de seqüència de l'execució de Record Linkage.

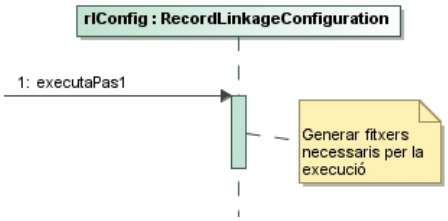


Figura 5.4: Diagrama de seqüència de la funció *executaPas1()*.

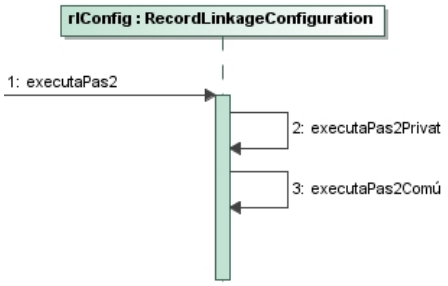


Figura 5.5: Diagrama de seqüència de la funció *executaPas2()*.

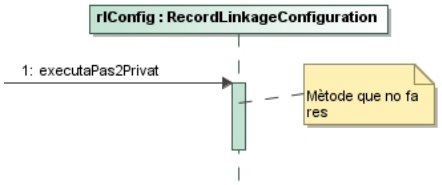


Figura 5.6: Diagrama de seqüència de la funció *executaPas2Privat()*.

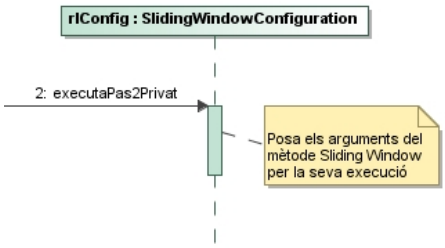


Figura 5.7: Diagrama de seqüència de la funció *executaPas2Privat()* redefinida pel mètode *Sliding Window*.

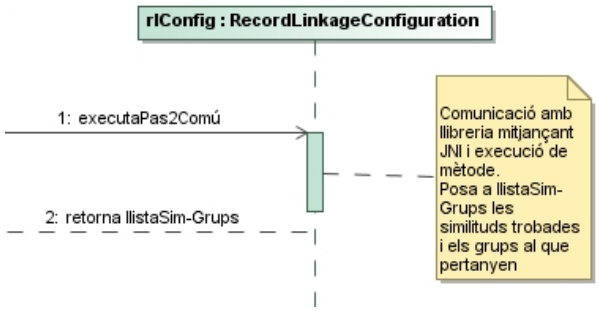


Figura 5.8: Diagrama de seqüència de la funció *executaPas2Comú()*.

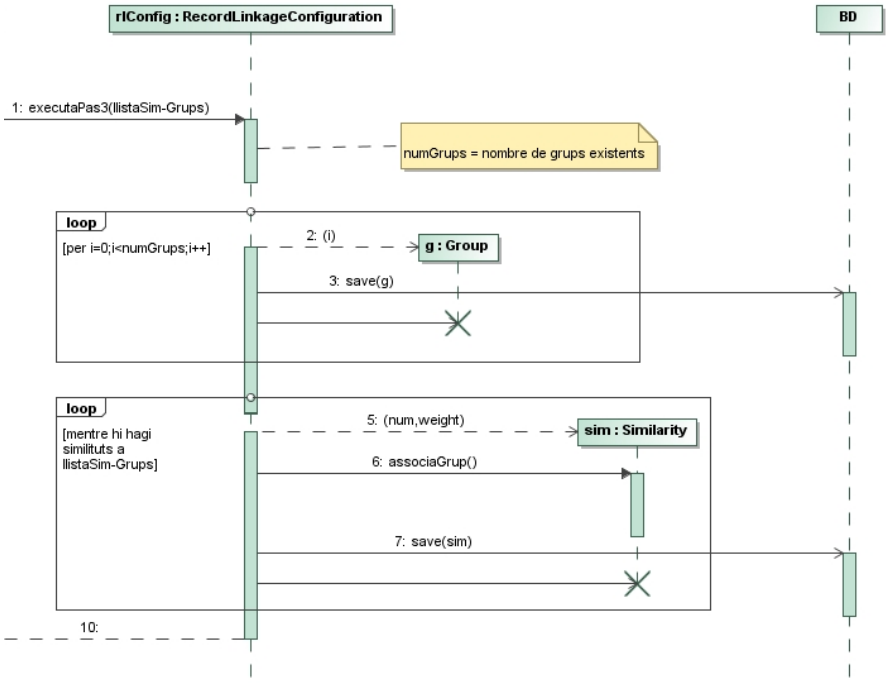


Figura 5.9: Diagrama de seqüència de la funció *executaPas3()*.

Com es pot observar, aquesta funcionalitat està dividida en tres passos: *executaPas1()*, *executaPas2()* i *executaPas3()*. El primer d'ells fa un preprocés necessari per a l'execució del mètode, el segon posa els paràmetres concrets de cada mètode de comparació i executa el mètoda i el tercer i últim que serveix per llegir els resultats en forma de grups i similituds i inserir-los a la base de dades. Recordem que l'execució la fa una llibreria feta en C++ i la qual es comunica amb Java mitjançant la tecnologia JNI i, per tant, no se n'ha fet el seu disseny.

El tercer model de comportament que es mostrarà és el referent a la graella de revisió. Tota similitud trobada durant el procés d'execució del mètode de Record Linkage té un estat, el qual pot variar entre *acceptada*, *rebutjada* i *desconeguda*. Així, els casos d'us Revisió per similituds i Revisió per grups estan realment formats per un conjunt de canvis d'estat de revisions decidits per part de l'usuari revisor. Aquest tercer diagrama de seqüència mostra doncs com funciona el canvi d'estat d'una sola similitud, de manera que el canvi d'estat de més d'una d'elles a la vegada suposa un lleu canvi en el disseny afegint-hi un bloc iterador. La Figura 5.10 mostra aquest comportament.

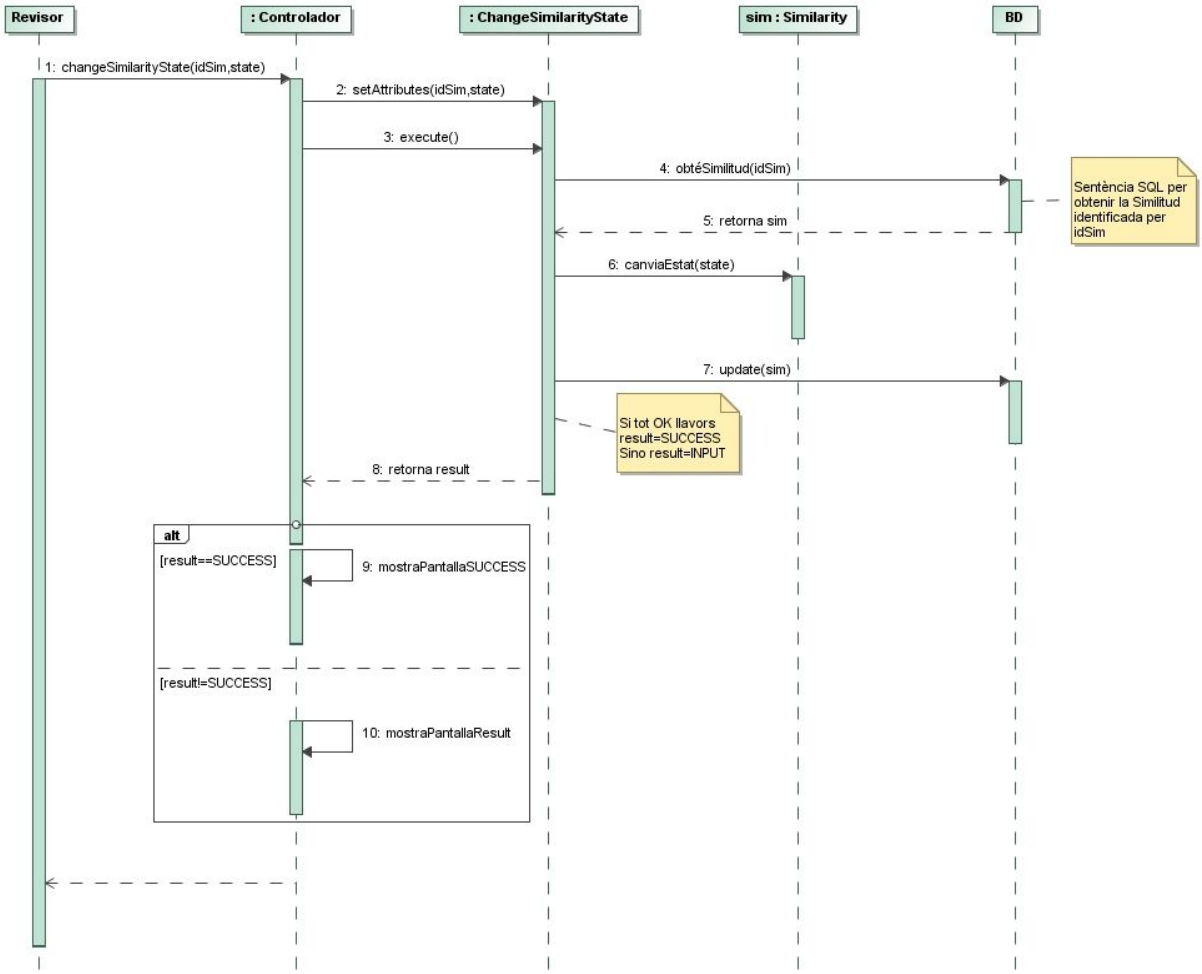


Figura 5.10: Diagrama de seqüència del canvi d'estat d'una similitud.

Es pot observar en aquest darrer diagrama que quan l'usuari decideix canviar l'estat d'una similitud, la qual té un identificador únic, el Controlador executa l'*action* corresponent a aquesta petició (URL) i es fa una consulta a la base de dades de la similitud i se'n modifica el seu estat.

5.3.2 Base de dades

Un altre punt bàsic en aquest projecte és l'emmagatzamament de la informació a base de dades. *JDBC* és una *API* de Java que serveix per a connectar-se a una base dades independentment del tipus de base de dades que s'estigui utilitzant i que, per tant, facilita l'accés a una base de dades als programadors. Avui en dia, la major part de les bases de dades són relacionals i, tal i com el seu nom indica, es basen en relacions, concepte força diferent al que és la orientació a objectes, la qual aporta abstracció, poliformisme i herència. Aquesta diferència entre tecnologies complica que una aplicació pugui fer persistents els seus objectes en la base de dades. Per a simplificar aquest mecanisme han aparescut *frameworks* de persistència d'objectes basats en un mapeig objecte-relacional o també coneguts com a *ORM*. Es tracta d'una tècnica de programació per a convertir dades entre el sistema de tipus utilitzat en un llenguatge de programació orientat a objetes com és Java i l'utilitzat en una base de dades relacional. En la pràctica això crea una base de dades orientada a objectes virtual, sobre la base de dades relacional. Aquest fet possibilita l'ús de les característiques pròpias de la orientació a objectes com són l'herència i el polimorfisme.

D'entre els *ORM* existents, s'ha decidit utilitzar el *framework* d'*Hibernate* [12], ampliament utilitzat en Java i que pot generar sentències SQL complexes. Per a utilitzar *Hibernate* s'han de crear els objectes de negoci i les seves relacions, les quals ja s'han decidit a la hora de crear el model conceptual durant l'especificació del sistema. Després s'haurà de crear un conjunt d'arxius XML que indicaran quin objecte es guarda en quina taula, indicant la relació entre propietats de l'objecte i columnes de la taula i també un altre arxiu on s'especifiqui la configuració de la connexió que ha d'utilitzar *Hibernate* per a accedir a la base de dades. Una vegada realitzats aquests pasos, es podrà indicar al *ORM* que persisteixi un objecte amb una simple crida com és:

orm.save(myObject);

Això generarà automàticament tot el codi SQL necessari per a emmagatzemar l'objecte. També existiran altres crides com són *load*, que permetrà carregar un objecte o *delete*, que l'esborrarà de base de dades.

Una vegada explicat el funcionament d'*Hibernate*, a continuació s'exposa un dels mapejos que s'han construit per a així veure el que significa emmagatzemar a base de dades.

```
<class name="DataSourceConfiguration" table="DATASOURCE_CONFIGURATION" >
        <id name="identifier" column="DSCONFIG_ID" type="integer">
                <generator class="native"/>
        </id>
        <property name="description" column="DS_CONFIG_DESCRIPTION" type="string" />
        <property name="createdBy" column="DS_CONFIG_CREATEDBY" type="string" />
        <property name="creationDate" column="CREATION_DATE" type="java.util.Date" update="false" not-null="true" />
        <list name="dataSource" table="DS_CONF_DATASOURCE" cascade="delete" lazy="false">
                <key column="DS_CONFIG_NAME" />
                <list-index column="DS_ID"></list-index>
                <many-to-many column="DATASOURCE" class="DataSource" />
        </list>
</class>
```

Aquest mapeig servirà per a emmagatzemar una configuració de fonts de dades (DataSourceConfiguration). Com es pot observar, constarà d'un identificador (*id*), el qual serà generat automàticament i tindrà com a nom *identifier*. També tindrà altres atributs representats amb l'etiqueta *property* com són una descripció, un camp on s'emmagatzemarà l'usuari que l'ha creat i un per la data en que s'ha creat. Finalment tindrà un element especial que és una llista de fonts de dades, el qual tindrà relació amb la classe *DataSource*. Totes les dades es guardaran a una taula anomenada *DATASOURCE_CONFIGURATION*, excepte la llista, la qual estarà a la taula *DS_CONF_DATASOURCE* per poder així guardar les relacions entre fonts de dades i configuracions.

A la Figura 5.11 es mostra la transformació que hi ha hagut des del model conceptual fins a poder aquestes dades a la base de dades.

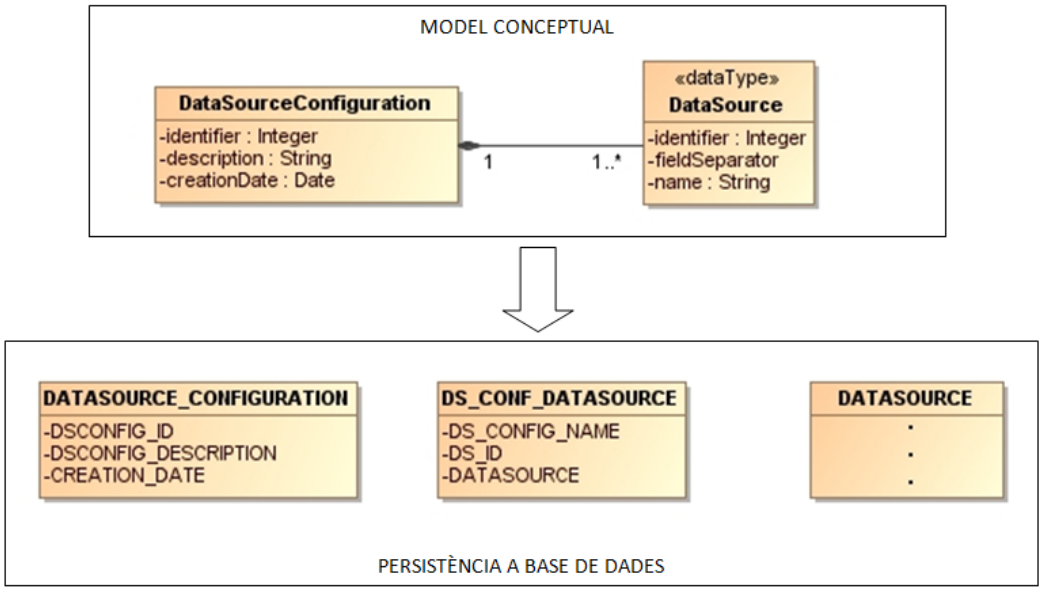


Figura 5.11: Transformació de model conceptual a persistència a base de dades utilitzant *Hibernate*.

Com es pot observar, s'ha passat de tenir dues classes a tres taules. Tot i així, pel que fa a la codificació, el programador no en serà conscient d'aquest canvi, ja que serà *Hibernate* qui generi el codi SQL necessari per a obtenir les dades, de manera que quan es vulguin obtenir les fonts de dades d'una configuració es farà una consulta sobre diferents taules. A més, degut a que només s'ha posat com a exemple un fragment del mapeig corresponent a la configuració de fonts de dades, veiem que la taula *DATASOURCE* no té cap columna, degut a que es desconeix el que conté només observant aquest fragment.

D'aquesta manera, s'ha explicat breument com es fa la persistència de dades en aquest projecte mitjançant un petit exemple que fa més fàcil la seva comprensió. Si es desitja observar tot el mapeig dels objectes a base de dades, es pot consultar l'Apèndix B, en el qual estan presents tots aquests elements.

5.4 Vista

La **Vista** és la part encarregada de construir la part visual de l'aplicació per a que els usuaris puguin interectuar-hi. Tal i com s'ha explicat anteriorment, el sistema està pensat per a ser executat en un entorn web i la interfície visual serà mostrada en codi *HTML*.

Les pàgines JSP, explicades al Capítol 5.2, permeten generar dinàmicament codi *HTML* incorporant *scripts* en *Java*. Així, en una pàgina JSP es pot tenir codi *HTML* estàtic i també dinàmic utilitzant etiquetes pròpies d'aquesta tecnologia o fins i tot creant-ne de noves. El que rep finalment l'usuari és simplement codi *HTML* típic, ja que és el servidor qui s'encarrega d'interpretar aquest codi dinàmic i en genera un d'estàtic que rebrà l'usuari.

Una altra tecnologia de la que se'n farà ús en aquesta part és la de *Tiles 2* [5]. Aquesta permet crear un sistema de plantilles de visualització per a una aplicació web com la que s'està construint. Es força comú que totes les pàgines que composen una aplicació web tinguin una part compartida, com poden ser els menús, la informació de login, les imatges de fons o els peus de pàgina i aquí és on ens podem ajudar creant plantilles. Així, les *tiles* permet definir una part comuna i la part canviant per a cada pàgina que es vol incorporar al sistema. Per a definir les plantilles i com es construiran les pàgines de l'aplicació s'utilitza un arxiu XML que és fàcilment editable.

Recordem que el funcionament de Struts 2 és el següent: es fa una petició URL, la qual es mapeja amb una

action, s'executa aquesta unitat de treball i finalment retorna com a resultat un String. El Controlador mostrarà una JSP o una altra segons el resultat retornat. Amb la introducció de la tecnologia de *tiles*, el controlador ja no mostrarà una JSP sinó que una *tile*, la qual estarà composada per una o vàries pàgines JSP.

A la Figura 5.12 es mostra el funcionament bàsic de *Tiles 2*. A dalt veiem l'arxiu XML que està composat per dues definicions: la primera anomenada *plantilla* defineix una plantilla, la qual utilitza la pàgina *plantilla.jsp* i la segona és la definició d'una acció concreta anomenada *acció1*, la qual extén la *plantilla* i modifica un dels seus valors per *accio1.jsp*. Sota aquest arxiu de configuració trobem què és el que contenen les dues JSP i el resultat que es mostraria si el Controlador mostra la *tile acció1*. Per a més informació sobre com funcionen les accions (o *actions*) es pot consultar el Capítol 5.2.

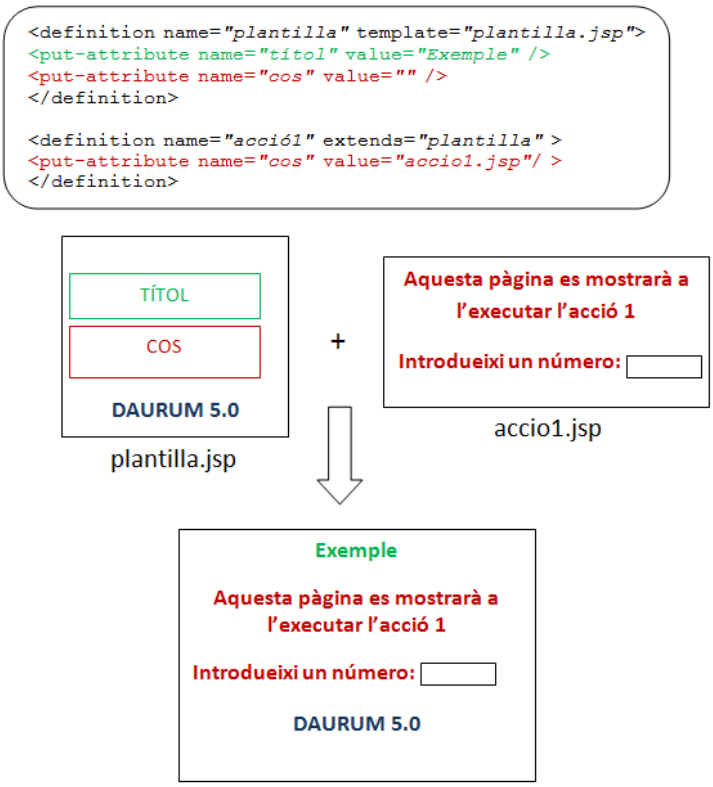


Figura 5.12: Exemple sobre el funcionament de *Tiles 2*.

És important comentar també que les pàgines contenen un alt índex de *JavaScript*. Aquest és un llenguatge de *scripting* interpretat pel navegador web que permet el desenvolupament d'interfícies d'usuari millorades i dinàmiques. En el cas d'aquest projecte, aquest llenguatge s'ha utilitzat per a embellir i modernitzar les pàgines i, per tant, fer-les més amigables pels usuaris.

JQuery [2] és una biblioteca o *framework* de *JavaScript* que permet simplificar la manera d'interactuar amb els documents HTML, manegar events, afegir efectes visuals i agregar interacció amb la tecnologia *Ajax* a les pàgines web. Així doncs, ens servirà per a oferir una sèrie de funcionalitats que requeririen de molt més codi si es volguessin construir des de zero amb *JavaScript*.

Exemples de la utilització de *JQuery* són les barres de progrés de l'aplicació, totes les taules presents, ja que permeten moltes interacció o tota la tecnologia *Ajax* utilitzada en la previsualització de fonts de dades.

Per últim, s'ha de fer referència a les classes introduïdes al Capítol 5.3 i que hem anomenat *preparers*. Les pàgines JSP poden rebre informació d'aquestes classes accedint als seus paràmetres. D'aquesta manera, obtenim pàgines

dinàmiques que dependran dels valors que tinguin els *preparers* en cada moment. A l'arxiu XML de definicions de les *tiles* es pot un atribut *preparer*, on s'especificarà quina serà la classe a executar i consultar per part de la JSP.

A continuació, a la Figura 5.13 es mostra un exemple del seu funcionament, on observem una classe *preparer* amb una llista com a atribut, un fragment de pàgina JSP amb unes etiquetes especials que permeten iterar sobre l'element accedit gràcies a una llibreria anomenada *Displaytag* i finalment la visualització que es tindria de la pàgina en el navegador web. A més, s'ha afegit un script de *JQuery* per a dotar a la taula d'efectes visuals.

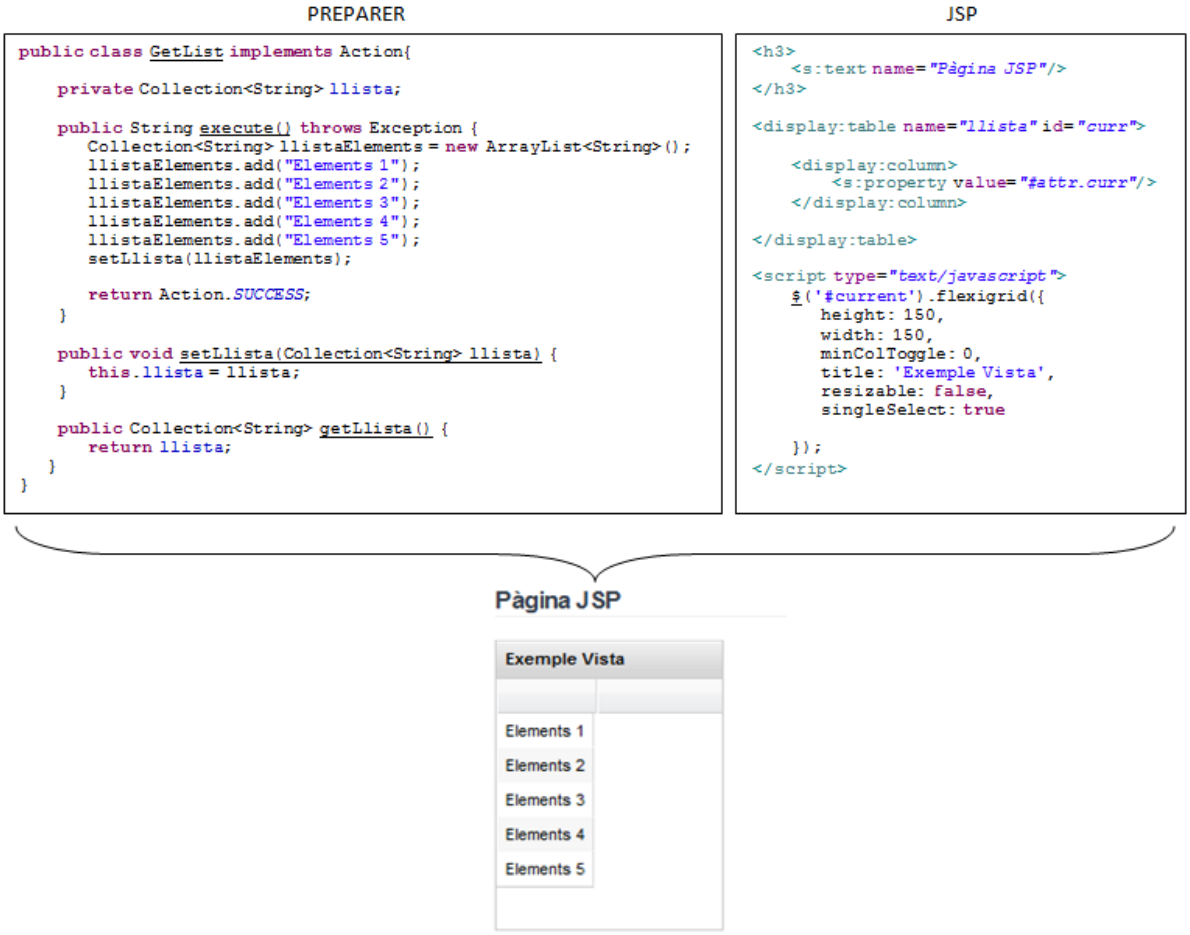


Figura 5.13: Exemple del funcionament *preparer*-JSP.

En aquest exemple, la iteració s'ha fet sobre un tipus bàsic com és *String*, però també és permès fer-ho amb objectes, podent així accedir a tots els seus atributs, de manera que dóna un gran ventall de possibilitats a la Vista.

5.5 Controlador

El Controlador és l'encarregat del fluxe de l'aplicació. Quan es fa una petició per part d'un usuari, ell la redirigeix a la *action* que executa la funcionalitat desitjada i finalment rep un resultat amb el qual es decideix mostrar una o altra pàgina. A part d'això, també s'executen els **interceptors**. Aquests seran uns o altres depenen de la petició que faci l'usuari, de manera que quan es faci el mapeig URL-*action* també es definiran els interceptors que s'executaran.

Els interceptors serveixen per a fer un preprocés i un postprocés sobre la *action* que s'està executant. Així doncs, abans i després de la *action* s'executaran una sèrie d'interceptors, també conegut com una **pila d'interceptors**, cadascun d'ells amb una tasca concreta. Existeixen també piles per defecte que permeten alleugerir la tasca d'assignar un grup d'interceptors a cada *action*.

A la Figura 5.14 veiem com s'executa aquesta pila d'interceptors i tot el funcionament general de Struts 2 quan hi ha una petició d'usuari. A més, observem també que els interceptors tenen ordre i, per tant, s'executen conforme han estat col·locats a la pila.

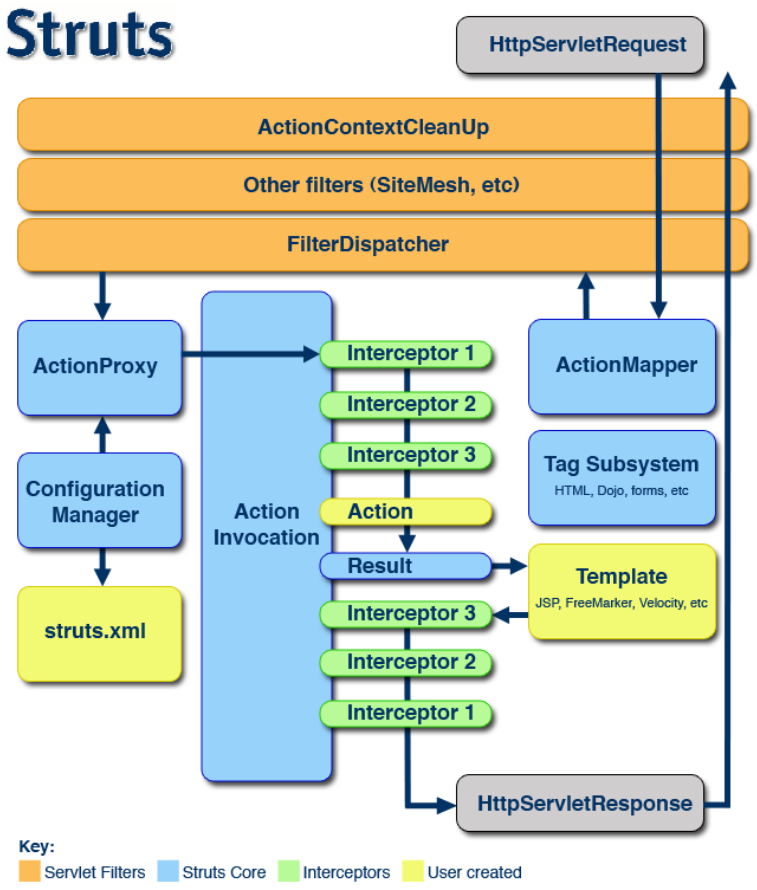


Figura 5.14: Ordre dels interceptors dins d'una petició d'usuari.

Ara és moment d'explicar quines tasques realitzen els interceptors que intervindran en aquest projecte, ja que sense descriure-la és difícil entendre el seu funcionament. Struts 2 proporciona una sèrie d'interceptors, però també se n'ha hagut de crear-ne de nous per a poder tenir controls més específics per a aquest projecte. Així doncs, a continuació mostrarem una llista amb els diferents i més importants interceptors utilitzats, alguns d'ells en totes les *actions* i altres només en unes concretes.

- Exception. Mapeja les excepcions cap a un resultat.

- Params. S'encarrega d'agafar els paràmetres d'una petició URL i posar-los com a atributs a la *action* que té mapejada.
- Workflow. Crida al mètode *validate()* de les *actions* abans d'executar-les. Així, permet que es faci una validació dels paràmetres de la petició i que anteriorment han estat afegits a la *action* gràcies a l'interceptor *Params*.
- HibernateAudit. Permet afegir auditoria a l'aplicació, de manera que es té constància dels canvis realitzats per cada usuari.
- StrutsMenu. Permet controlar quins elements del menú es mostraran a l'usuari en funció dels rols que posseeixi.
- ExecAndWait. Permet posar una *action* en *background*, de manera que mentre s'està executant es pot mostrar a l'usuari la barra de progrés.
- Roles. S'encarrega de controlar si un usuari té permís per a executar la funcionalitat desitjada consultant els seus rols.
- Hibernate. És l'encarregat d'iniciar una transacció per a cada petició i fer-ne un *rollback* en cas de que s'hagi llençat alguna excepció.

Com es pot observar, les funcionalitats dels interceptors poden ser molt diverses, podent ser referents a tasques relacionades amb l'emmagatzemament com *Hibernate* fins a relacionades amb la Vista com *StrutsMenu*.

A continuació, a la Figura 5.15, es mostra la manera en que es defineix el mapeig d'una certa petició amb una *action*, els interceptors que s'hauran d'executar i les diferents pàgines que s'hauran de mostrar en funció del resultat retornat per la *action*.

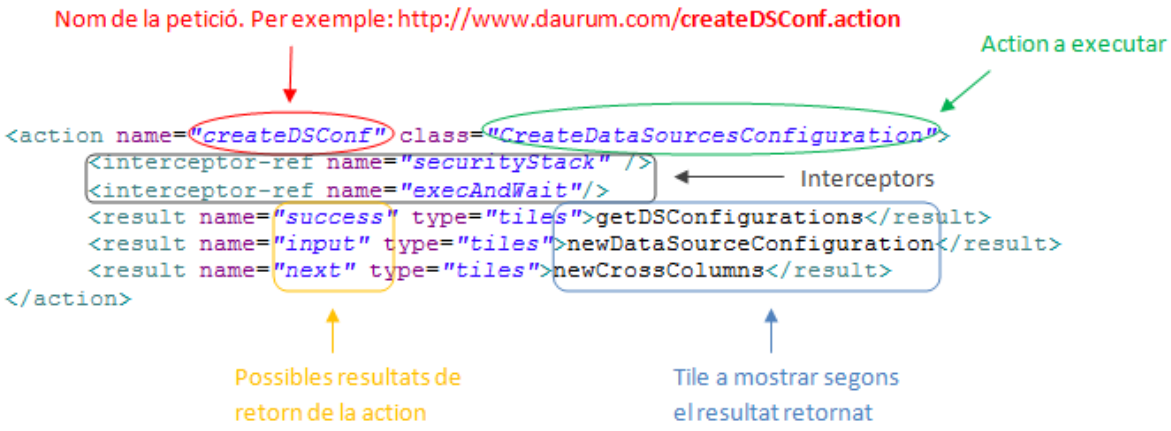


Figura 5.15: Mapeig d'una petició utilitzant Struts 2.

Es tracta d'un mapeig real que pertany a la petició de creació d'una configuració de fonts de dades i que té com a interceptors *securityStack* i *execAndWait*. De la segona ja ha estat descrita la seva funcionalitat i la primera engloba un conjunt d'interceptors entre els que destaca el control d'accés a la creació d'aquest tipus de configuració, ja que només un usuari expert pot fer-ho. Les *tiles* que es mostraran segons el resultat retornat estaran definides en un altre arxiu XML tal i com s'ha explicat al Capítol 5.4 i se'n mostra un exemple a la Figura 5.12.

5.6 Resum

El disseny del sistema a desenvolupar es divideix en tres parts ben diferenciades: el Model, la Vista i el Controlador.

El Controlador estarà format per un arxiu XML on es faran els mapejos de les peticions, les quals executaran una *action* i que tindrà un conjunt d'interceptors, que tindran la tasca de fer un preprocés i un postprocés sobre la funcionalitat accedida. També es definirà quina pàgina s'haurà de mostrar segons el resultat que hagi retornat l'execució de la *action*.

El Model estarà format principalment per les *actions*, que serveixen per a dotar d'una certa funcionalitat al sistema, de les classes de la lògica de negoci i de les necessàries per a aportar informació a la interfície d'usuari: *preparers*.

La Vista la conformaran totes les pàgines que podran visualitzar els usuaris mitjançant el seu navegador web i que seran JSP per a poder així accedir a la informació dels *preparers*. A més, incorporaran *JavaScript* per fer-les així més agradables per l'usuari i utilitzaran plantilles per a poder reutilitzar la visualització que no canvia.

Capítol 6

Testing

Una vegada realitzada tota la implementació del sistema utilitzant l'arquitectura dissenyada a la fase anterior, és moment de testejar-lo per a revelar la qualitat del software.

6.1 Proves de software existents

Les proves de software són els procesos que permeten verificar i revelar la qualitat d'un producte software. Són utilitzades per a identificar possibles errors en la implementació, qualitat o usabilitat i són, bàsicament, una fase del desenvolupament que consisteix en provar l'aplicació construïda. Es poden distingir proves de diferents tipus:

- Unitàries. Serveixen per a provar el funcionament de qualsevol mètode no trivial existent en el codi i per a assegurar que cadascun d'ells donarà un resultat correcte per separat.
- Funcionals. Serveixen per a provar que els casos d'ús de l'aplicació poden ser executats correctament.
- D'integració. Són executades una vegada aprobades les unitàries i únicament per a comprovar el correcte funcionament de tots els elements unitaris executats de forma conjunta.
- De validació. Són el procés de revisió pel qual es verifica que el sistema software compleix amb les especificacions i amb el que el client desitjava.
- De regressió. Serveixen per a descobrir nous errors, carències de funcionalitat o divergències funcionals respecte al comportament esperat del software, induïdes per canvis recents realitzats en parts de l'aplicació i que sense ells no apareixien.
- De rendiment. Serveixen per a determinar la rapidesa en que es realitza una tasca del sistema en determinades circumstàncies.

Veiem que existeixen molts tipus de proves a realitzar en un software [14]; però no sempre és convenient realitzar-les totes si el disseny utilitzat ja garanteix el correcte funcionament del sistema.

6.2 Proves realitzades i resultats obtinguts

En el sistema que s'ha desenvolupat, els tests realitzats no han estat exhaustius ni extensos de cada element degut a que es tracta d'un projecte acadèmic amb un calendari establert i s'ha optat per anar realitzant proves a mida que s'incorporaven noves opcions.

Recordem que el desenvolupament del projecte s'ha fet en espiral i això ha permès poc a poc anar incorporant noves funcionalitats. Així, cada vegada que s'han acabat d'implementar aquestes, se n'ha provat el seu funcionament amb un conjunt de dades utilitzat durant tot el projecte, de manera que si es detectava algun error es solucionava durant aquella fase abans de passar a la següent, on es determina quines noves funcionalitats es poden afegir.

Pel que fa a les proves de validació, aquestes s'han dut a terme mitjançant reunions amb el Director d'aquest projecte, el qual actua com a cilent. Durant aquestes reunions s'ha mostrat el funcionament de l'aplicació, s'ha validat que els requeriments eren els demanats i s'ha decidit quins elements es podien millorar.

Respecte al funcionament de les pàgines web que conformen el sistema, s'han testejat introduint tota mena de valors nuls o invàlids per a detectar possibles errors i així controlar-los per a que el sistema mostri un missatge d'error en cas d'existir. Com s'ha comentat al Capítol 5.5, els interceptors han ajudat a fer aquesta feina, gràcies a que permeten fer un preprocés abans d'executar una funcionalitat i així es pot detectar si els valors entrats per l'usuari són incorrectes.

En canvi, sí que pot ser útil realitzar proves de rendiment de l'aplicació. És per això que una vegada finalitzada la implementació del projecte, s'han realitzat un conjunt d'execucions de Record Linkage per a comprovar que el funcionament era el correcte. Això ha permès detectar i solucionar fugues de memòria i problemes relacionats amb la base de dades.

A continuació, es mostren alguns resultats obtinguts en les execucions. Pel primer exemple, s'han provat d'executar unes mateixes fonts de dades amb diverses mides de finestra i han permés observar un resultat esperable: quan més grossa la finestra, més triga en fer-se l'execució perquè més comparacions s'han de fer i també que, arribat a un cert valor de finestra, és improductiu provar amb un valor més gran ja que poques noves similituds es trobaran.

Així, la Figura 6.1 mostra una gràfica amb els resultats obtinguts d'execucions amb dos fonts de dades de mil registres cadascuna i tres columnes de creuament, de les qual dues eren claus d'ordenació. La mida de la finestra es pot observar en la gràfica i es representa amb W.

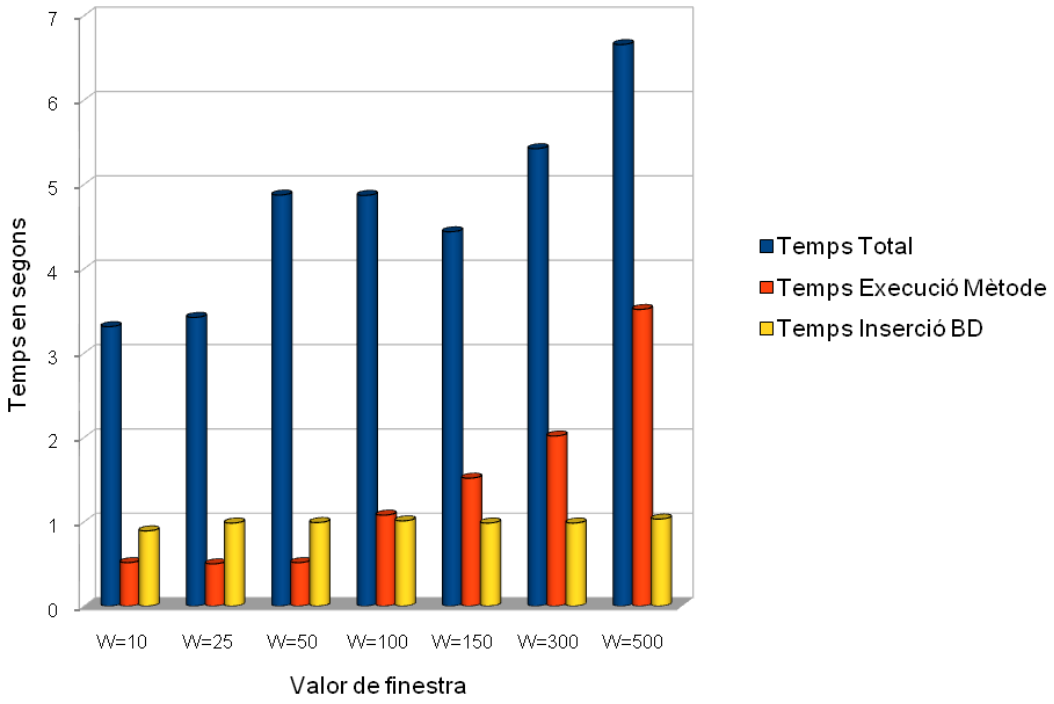


Figura 6.1: Algunes execucions realitzades durant la fase de testing.

Com es pot observar, el temps d'execució de les comparacions (Temps Execució Mètode) va augmentant de forma linial a mida que s'incrementa el valor de la finestra. Una vegada trobades les similituds, aquestes s'han d'incorporar a la base de dades i, per tant, també és un temps que s'ha de tenir en compte. A l'exemple anterior, veiem que aquest temps s'ha mantingut constant i només ha augmentat lleument entre una finestra de 10 i una de 25. Això és degut a que el nombre de similituds trobades, i per tant a inserir a la base de dades, ha estat semblant amb tots els valors de finestra, excepte per la finestra amb valor 10, en el qual ha estat lleument inferior. A continuació mostrem una taula amb el nombre de similituds trobades en cada cas per a poder comprovar aquest fet i el temps que s'ha trigat per a inserir-les.

	W=10	**W=25**	**W=50**	**W=100**	**W=150**	**W=300**	**W=500**
Nombre Similituds	187	220	232	232	233	234	234
Temps Inserció BD (segons)	0,891	0,984	0,99	1,01	0,984	0,984	1,031

Una altra prova que s'ha realitzat ha estat la de variar el percentatge mínim d'execució per a una finestra prefixada i les mateixes fonts de dades utilitzades anteriorment. Els resultats es mostren al quadre següent.

	>90%	**>80%**	**>70%**	**>60%**	**>50%**	**>25%**	**>10%**
Temps Total	3,72	3,879	4,137	4,728	5,328	28,384	75,559
Temps Execució Mètode	0,515	0,516	0,956	1,009	1,016	1,518	1,519
Temps Inserció BD	0,699	0,85	0,891	1,251	2,104	24,521	71,735
Temps Generació fitxers auxiliars	2,506	2,513	2,29	2,468	2,208	2,345	2,305
Nombre Similituds trobades	78	93	117	245	609	9318	28743

Com és d'esperar, a mida que acceptem similituds amb un pes més baix, els temps total d'execució va en augment. El que més sorpren, és que el temps més rellevant el marca la inserció dels resultats a la base de dades a mida a partir d'un cert nombre de similituds trobades. Això és així, perquè les execucions s'han realitzat en un ordinador personal i amb una base de dades poc potent implementada en Java i, per tant, el temps augmenta quan es necessiten inserir moltes dades. Pel que fa a l'execució del mètode de comparació, ha augmentat de forma lleu, degut a que el nombre de comparacions entre registres serà el mateix per a totes les execucions i variant només degut al procés que s'ha de fer per a calcular el pes entre dos registres (el qual no s'acaba de realitzar quan ja se sap que no podrà superar el llindar establert per l'usuari). Per últim, el temps que no varia significativament és el de generació de fitxers auxiliars, degut a que ve donat per les fonts de dades, de les que ja s'ha comentat que s'han utilitzat les mateixes per a totes les execucions.

Així doncs, una vegada realitzades algunes proves de rendiment, es pot concloure que la qualitat de la base de dades utilitzada influirà força en els temps d'execució del Record Linkage i farà que es puguin obtenir els resultats de forma més ràpida.

Capítol 7

Planificació i costos

Després d'haver descrit exhaustivament el sistema que es vol construir, veient la seva especificació i disseny, és moment de presentar la planificació i la viabilitat econòmica en les quals caldrà recolzar-se per a poder dur a terme el seu desenvolupament. En l'àmbit de l'enginyeria és vital tenir organitzades les tasques d'un procés, ja que si no fós així seria impossible implementar una aplicació en els límits temporals i econòmics establerts pel client.

7.1 Planificació del projecte

La planificació d'un projecte no és una tasca que es realitzi en un moment concret i ja no es torni a modificar, sinó que comença des del primer dia en que sorgeix el projecte i dura fins al lliurament final. Descriu l'organització que es durà a terme, l'assignació de recursos i la duració de les tasques a desenvolupar durant el cicle de vida de la construcció d'un sistema software.

La planificació a l'inici d'un projecte és una previsió de temps de cada tasca que forma part del sistema que es vol construir. A partir d'ella sorgeix un pressupost que marca la viabilitat econòmica i el futur del projecte, ja que serà presentat al client i aquest decidirà si hi està interessat. Aquesta primera planificació, que es fa en base a l'experiència adquirida en altres projectes acabats, canvia necessàriament a mesura que passa el temps, degut a que ha d'anar adaptant-se a les desviacions comeses durant el decurs del projecte. D'aquesta manera, aquesta planificació acaba resultant una descripció real de les hores que s'han destinat a cada tasca i permet calcular els seus costos finals.

Per tal de simplificar l'extensió d'aquest capítol, no es mostraran les diverses versions de la planificació feta en aquest projecte, sinó que s'exposarà el resultat obtingut després d'haver finalitzat el projecte. També cal dir que hi ha diverses tasques que es realitzaran en paral·lel mentre que d'altres ho faran en seqüència, degut a les depèndencies que existeixen entre elles.

La planificació i costos han estat desenvolupats i calculats mitjançant la potent eina de gestió de projectes anomenada *Microsoft Project*.

7.1.1 Diagrama de Gantt

Els diagrames de Gantt permeten representar de forma visual el calendari d'un projecte i els terminis i dependències existents entre les tasques que el conformen. A més, es habitual englobar diverses tasques en una de més extensa per a així veure els punts claus en els que es divideix un projecte.

En aquest projecte, els punts claus seran les diferents funcionalitats que permet utilitzar, entre les que destaquen la gestió d'usuaris, la gestió de configuracions per al Record Linkage, l'execució del mètode, la revisió i la posterior exportació de dades. A més, la construcció del model conceptual i la seva implementació ocuparan un lloc vital en la planificació del projecte, ja que constantment estarà modificant-se per a adaptar-se a les funcionalitats que s'aniran afegint a mesura que se n'afegeixin de noves. És per això, que la implementació i integració amb la base de dades del model conceptual abarcarà gairebé tota la durada del projecte.

La formació que s'haurà de dur a terme per a conèixer el funcionament de les diferents tecnologies també és un punt que s'ha d'incorporar a la planificació de tot projecte. A més, també s'han contemplat les diferents fites presents en aquest PFC, com poden ser l'entrega de l'informe previ o la matriculació del projecte, les quals es marquen en un diagrama de Gantt amb un rombe i la data escollida.

Amb tot això, a la Figura 7.1 observem la planificació realitzada, on es contemplen aquests diversos punts comentats anteriorment.

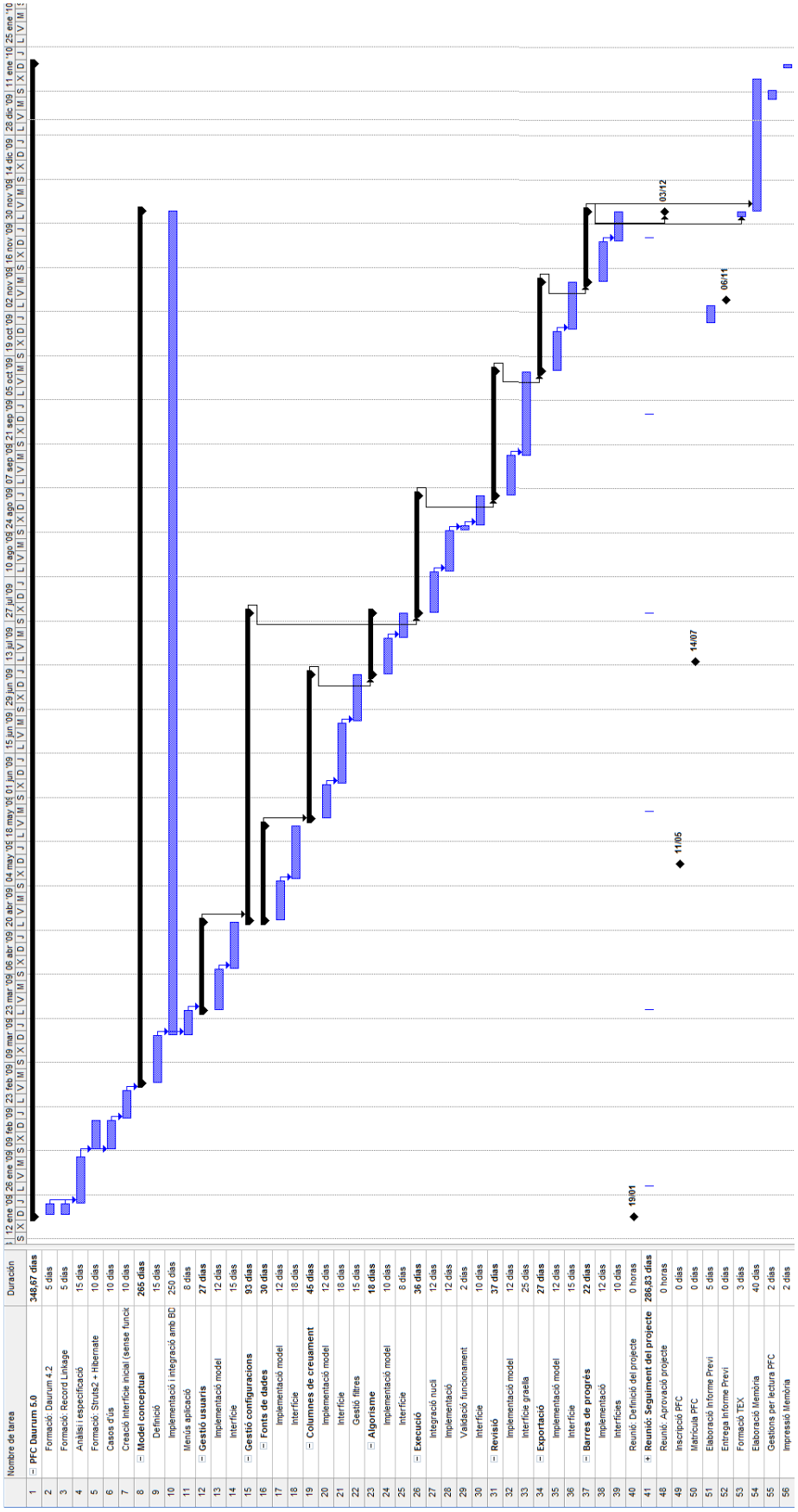


Figura 7.1: Diagrama de Gantt obtingut a la finalització del projecte.

7.2 Costos del projecte

Tal i com ja s'ha fet èmfasi amb anterioritat, és molt important que la planificació d'un projecte vagi modificant-se constantment, degut als desviaments que poden sorgir durant el seu decurs, però també trascendental és haver-ne fet una d'inicial, d'on pugui mesurar-se la seva viabilitat econòmica amb l'objectiu de determinar un pressupost. Amb aquest, es podrà comprovar que hi hagi suficients recursos per tal fer-ne possible la seva realització en el temps marcat.

Ara bé, quan ja ha finalitzat el projecte, és necessari anotar els costos finals per a determinar l'import final que tindrà el projecte i que, per tant, el client haurà de pagar.

En aquest projecte i, en la majoria de projectes, hi intervenen dos tipus de recurs. Per un costat, els recursos humans, als quals se'ls hi encarregaran tasques a realitzar i que, per tant, són els que ajudaran a dur-lo a terme. Per l'altre, els recursos materials que són eines que ajuden a fer possible la realització de les tasques que conformen el projecte

En la configuració del cost total del projecte solen intervenir-hi també altres factors, com per exemple despeses per desplaçaments, utilització del sistema elèctric, desamortització de les eines utilitzades, imprevistos... Aquest factors no es tindran en compte en el càlcul final dels costos del projecte, ja que resulten més complexes de mesurar.

7.2.1 Recursos Humans

En aquest projecte, els costos que representen un volum més important del conjunt de despeses són els que fan referència a les persones que hi estan involucrades. El fet de que es tracti d'un projecte de final de carrera implica que totes les tasques han estat realitzades per un becari de projectes recolzant-se en la figura del Director de projecte i, per tant, no ha existit un grup de persones especialitzades en cadascun dels punts que s'hagin repartit la feina a fer.

Ara bé, per a mostrar un càlcul dels costos associats al projecte més realista s'ha cregut oportú fer una simulació d'un grup de recursos humans, als quals se'ls hi assignarà les diverses tasques explicitades en el diagrama de Gantt. No obstant, les hores destinades a la realització de cada tasca són totalment verídiques.

El conjunt de persones implicades seran: un analista, un dissenyador, un programador i un formador. En aquest projecte en concret, s'ha decidit no tenir en compte la figura del cap de projecte i les seves tasques se les repartiran entre l'analista i el dissenyador.

L'analista i el dissenyador seran els encarregats de fer l'anàlisi, especificació i disseny de l'aplicació juntament amb tota la documentació necessària. El programador serà el que durà a terme les decisions preses per ells i, per tant, s'encarregarà de la codificació de l'aplicació. Per últim, la figura del formador servirà per a ensenyar les noves tecnologies que s'utilitzin en el projecte. A continuació, a la Figura 7.2, es mostra el barem de preus per hora fixats per a cada rol implicat.

Nom del recurs	Tarifa estàndar
Analista	35 €/hora
Dissenyador	30 €/hora
Formador	20 €/hora
Programador	18 €/hora

Figura 7.2: Tarifes fixades pels treballadors ficticis.

A partir del calendari laboral obtingut del diagrama de Gantt i tenint en compte el barem de tarifes escollit, s'ha assignat a cada tasca un o diversos treballadors i ha estat possible determinar el cost total del projecte. A la Figura 7.3 es poden observar aquestes assignacions, el cost que té la realització de cada tasca i el cost final del projecte.

Identificador	Nom de la tasca	Número de Dies	Recursos assignats	Hores	Cost
1	**PFC Daurum 5.0**	**348,67 dies**		**2064h**	**49.430,00 €**
2	Formació: Daurum 4.2	5 dies	Formador [50%]	12,5h	250,00 €
3	Formació: Record Linkage	5 dies	Formador [50%]	12,5h	250,00 €
4	Anàlisi i especificació	15 dies	Analista [50%], Dissenyador [50%]	37,5h + 37,5h	2.437,50 €
5	Formació: Struts2 + Hibernate	10 dies	Formador	50h	1.000,00 €
6	Casos d'ús	10 dies	Analista [50%], Dissenyador [50%]	25h + 25h	1.625,00 €
7	Creació Interfície inicial (sense funcionalitats)	10 dies	Analista [40%], Dissenyador [40%], Programador [20%]	16h + 16h + 8h	1.184,00 €
8	Model conceptual	265 dies			
9	Definició	15 dies	Analista [50%], Dissenyador [50%]	37,5h + 37,5h	2.437,50 €
10	Implementació i integració amb BD	250 dies	Programador [20%]	250h	4.500,00 €
11	Menús aplicació	8 dies	Analista [50%], Dissenyador [50%]	20h + 20h	1.300,00 €
12	Gestió usuaris	27 dies			
13	Implementació model	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
14	Interfície	15 dies	Programador	75h	1.350,00 €
15	Gestió configuracions	93 dies			
16	Fonts de dades	30 dies			
17	Implementació model	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
18	Interfície	18 dies	Programador	90h	1.620,00 €
19	Columnes de creuament	45 dies			
20	Implementació model	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
21	Interfície	18 dies	Programador	90h	1.620,00 €
22	Gestió filtres	15 dies	Analista [33%], Dissenyador [33%], Programador [33%]	25h + 25h + 25h	2.075,00 €
23	Algorisme	18 dies			
24	Implementació model	10 dies	Analista [25%], Dissenyador [25%], Programador [50%]	12,5h + 12,5h + 25h	1.262,50 €
25	Interfície	8 dies	Programador	40h	720,00 €
26	Execució	36 dies			
27	Integració nucli	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
28	Implementació	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
29	Validació funcionament	2 dies	Analista [50%], Dissenyador [50%]	5h + 5h	325,00 €
30	Interfície	10 dies	Programador	50h	900,00 €
31	Revisió	37 dies			
32	Implementació model	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
33	Interfície graella	25 dies	Programador	125h	2.250,00 €
34	Exportació	27 dies			
35	Implementació model	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
36	Interfície	15 dies	Programador	75h	1.350,00 €
37	Barres de progrés	22 dies			
38	Implementació	12 dies	Analista [25%], Dissenyador [25%], Programador [50%]	15h + 15h + 30h	1.515,00 €
39	Interfícies	10 dies	Programador	50h	900,00 €
40	Reunió: Definició del projecte	0 dies	Analista, Dissenyador, Programador		
41	Reunió: Seguiment del projecte	286,83 dies			
42	Reunió: Seguiment del projecte 1	1 hora	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
43	Reunió: Seguiment del projecte 2	1 hora	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
44	Reunió: Seguiment del projecte 3	1 hora	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
45	Reunió: Seguiment del projecte 4	1 hora	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
46	Reunió: Seguiment del projecte 5	1 hora	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
47	Reunió: Seguiment del projecte 6	1 hora	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
48	Reunió: Aprovació projecte	0 dies	Analista, Dissenyador, Programador	1h + 1h + 1h	83,00 €
49	Inscripció PFC	0 dies			
50	Matrícula PFC	0 dies			
51	Elaboració Informe Previ	5 dies	Analista [50%], Dissenyador [50%]	12,5h + 12,5h	812,50 €
52	Entrega Informe Previ	0 dies			
53	Formació TEX	3 dies	Formador	3h	60,00 €
54	Elaboració Memòria	40 dies	Analista [50%], Dissenyador [50%]	100h +100h	6.500,00 €
55	Gestions per lectura PFC	2 dies			
56	Impressió Memòria	2 dies			

Figura 7.3:Assignació de recursos i costos del projecte.

Es pot veure que el cost de personal del projecte és de 49.430 euros i el nombre d'hores dedicades a la seva construcció comptant les diferents fase (anàlisi, disseny i implementació) ha estat de 2.064.

7.2.2 Recursos Materials

A part dels costos referents al personal utilitzat per al desenvolupament del software, també és important afegir-ne els derivats de la utilització de maquinari i software que han estat necessaris per a assoliar aquesta tasca.

Es calcula que hauran estat necessaris tres equips informàtics amb els seus respectius perifèrics per a la realització d'aquest projecte i tenint en compte que han intervingut tres tipus de recursos diferents: un per l'analista, un altre pel dissenyador i un tercer pel programador. A més, pel que a les aplicacions que requereixen de llicència per a ésser utilitzats s'ha fet ús del sistema operatiu *Microsoft Windows XP* i el paquet software de *Microsoft Office 2003* per a poder fer la planificació del projecte i calcular-ne els costos. També s'ha adquirit una llicència estàndar del software *Magic Draw* que permetrà fer-ne un disseny i generar diagrames de l'aplicació a diferents nivells.

Equip informàtics	1.000 € x 3 = 3.000 €
Microsoft Windows XP	125 €
Microsoft Office 2003	50 €
Magic Draw Standard Edition	400 €
Total costos materials	**3.575 €**

7.2.3 Cost Total

Així doncs, una vegada s'han calculat els costos relacionats amb el personal i amb el material utilitzat, podem fer el càlcul final del cost del projecte, que serà la suma d'aquests dos apartats. Cal recordar que aquests costos són fictícis degut a que es tracta d'un projecte de final de carrera, però que la planificació de tasques i temps dedicats és totalment real.

Cost personal	49.430 €
Cost material	3.575 €
Cost Projecte	**53.005 €**

Capítol 8

Conclusions i Futur del projecte

Aquest darrer capítol té com a objectiu valorar tot el que m'ha aportat el desenvolupament d'aquest projecte tant a nivell personal com professiona. A més, tot i tractar-se d'un projecte acadèmic, és molt possible que en tingui una continuitat i, per tant, és important fer un anàlisi de les futures millores que se'n podrien incorporar.

8.1 Valoració

Els objectius fixats a l'inici del projecte s'han assolit amb escreix i es pot dir que s'ha aconseguit tenir una aplicació web per a Record Linkage llesta per a utilitzada.

El **disseny** proposat ha fet que s'aconsegueixi una arquitectura del software molt robusta i fàcilment reutilitzable per a la construcció de futures aplicacions.

Les **funcionalitats** de les que s'ha dotat a aquest sistema són totes les necessàries per a poder executar un Record Linkage. Permetrà a l'usuari afegir nous fitxers, configurar molt al detall els paràmetres d'execució per a aconseguir millors resultats i fer-ne una posterior revisió no gaire feixuga gràcies a les diverses opcions de canvi d'estats i visualitzacions de les similituds que s'incorporen. A més, la configuració de l'exportació també serà molt personalitzada per a cada execució que es dugui a terme. També s'ha afegit una gestió de filtres editable que permetrà a l'usuari estandaritzar el contingut dels fitxers d'entrada com més desitgi. Per últim, a nivell d'administrador, s'ha dotat d'una gestió d'usuaris amb diversos rols, de manera que existirà un gran control sobre l'aplicació.

Respecte a la **interfície gràfica**, s'ha intentat aconseguir una visualització molt acurada per a que els usuaris tinguin la sensació de que és fàcil utilitzar-la i s'ha cuidat molt cadascuna de les pàgines que la composen afegint-hi endemés elements tecnològics que permeten tenir més efectes visuals.

A nivell personal, em sento molt satisfet de la feina feta en els punts anteriorment esmentats, dels quals se n'ha intentat cuidar al màxim cadascun d'ells. El fet de desenvolupar el projecte dins d'un grup de recerca com és DAMA-UPC m'ha permès aprendre molt dels meus companys, tots ells amb un molt alt nivell de coneixement en les diferents matèries existents. Des del començament els membres del grup han estat oberts a qualsevol dubte que sorgia i m'han ajudat sempre que han pogut. A més, l'estància a DAMA-UPC m'ha permès decidir quins seran els següents passos a seguir una vegada obtingui la titulació a nivell professional, ja que m'ha servit per a conèixer un món laboral desconegut i molt diferent al experimentat en altres empreses privades que he tingut la oportunitat de treballar.

8.2 Coneixements previs

Tot i que el departament d'Arquitectura de Computadors es centra principalment en temes relacionats amb el hardware, el grup de recerca DAMA-UPC disposa d'un conjunt de gent que es dedica al desenvolupament de software. En concret, DAMA-UPC centra la seva investigació en l'eficiència i èxit en el tractament de grans volums de dades.

En aquest projecte es tenia la intenció de crear una nova aplicació per a Record Linkage que donés solucions als problemes trobats en l'anterior versió de Daurum. S'havia de dissenyar un sistema totalment nou, molt diferents a l'existent i del qual només se n'aprofitaria una llibreria per a l'execució de l'algorisme de RL i a més poder centralitza la seva informació de manera que tots els usuaris es puguessin aprofitar d'ella.

Així doncs, ha estat vital el coneixement adquirit en les assignatures relacionades amb l'enginyeria del software, com són **Enginyeria del Software 1** i **Engineria del Software 2** i les relacionades amb l'emmagatzemament de dades com és **Base de Dades**. Altres assignatures obligatòries de la carrera com són les de programació també han estat importants per a poder desenvolupar i finalitzar aquest projecte.

Respecte a les assignatures optatives cursades, els coneixements adquirits a **Programació Concurrent i Distribuïda** m'han ajudat a la implementació de *Threads*, **Programació Conscient de l'Arquitectura** m'ha permés aplicar certes tècniques estudiades i **Recuperació de la Informació** m'ha permès entendre el funcionament de l'algorisme del Record Linkage. També, per a la realització de la memòria han estat útils assignatures com **Presa de Decisions i Gestió de Projectes Empresarials**, **Planificació i Gestió de Projectes i Sistemes Informàtics** o **Tècniques i Processos de Gestió Empresarial**, les quals m'han dotat de coneixements sobre la gestió i planificació del projecte.

8.3 Coneixements adquirits

Una vegada finalitzat el projecte s'ha de dir que han estat molts els coneixements adquirits durant la seva realització. La major part d'ells han estat de caràcter pràctic tal i com pretén la realització d'un projecte final de carrera. A continuació es llista un resum d'aquests coneixements.

- Introducció al Record Linkage.
- Implementació de software amb *Java*.
- Utilització de l'*IDE* de *Eclipse*.
- Utilització de l'eina de gestió i construcció de projectes *Maven*.
- Coneixements en aplicacions web i en el funcionament de *Tomcat*.
- Utilització del *framework Struts 2*, que implementa el patró MVC.
- Utilització d'un *ORM* per a la persistència d'objectes com és *Hibernate*.
- Coneixements en generació de pàgines web mitjançant *JSP*.
- Utilització de *Tiles 2* per a fer plantilles de les pàgines web.
- Utilització de la llibreria *JQuery* per a dotar les pàgines de *JavaScript*.
- Utilització de Lyx [10] per a la generació de documents.
- Introducció al llenguatge de generació de documents LaTeX.

8.4 Treball futur

Degut a que és possible que el projecte tingui una continuitat i no es tracti merament d'un projecte de final de carrera, és important també definir els passos a seguir en un futur. És per això que a continuació es mostra una llista de possibles millores i noves funcionalitats que podrien aparèixer en noves versions i que suposarien un avanç en Daurum.

- Nous mètodes de comparació de registres. En aquest projecte només es dóna la possibilitat d'executar l'algorisme de *Sliding Window* i, per tant, és important donar un ventall més ample d'opcions als usuaris. A més, la llibreria existent ja permet l'execució del mètode de *Standard Blocking* i, recordem també que el disseny i implementació del sistema s'ha fet pensant en possibles futures inclusions de nous mètodes i, per tant, no seria complicat de fer-ho.

- Funcionament en tots els navegadors. Actualment aquesta aplicació web funciona perfectament en els navegadors *Mozilla Firefox, Internet Explorer* i *Google Chrome*. Seria important que funcionés en altres navegadors existents, els quals tenen especificacions diferents entre ells i que, a vegades, no compleixen les recomenacions fetes per *W3C [11],* Consorci Internacional per a la World Wide Web.

- Revisió de similituds intel·ligent. Una altra funcionalitat interessant seria la d'incloure una revisió intel·ligent de manera que, quan es canviés l'estat d'una similitud, el mateix sistema canviés l'estat d'altres similituds relacionades.

- Visualització en forma de graf. Les similituds estan formades sempre per dos registres i tenen un valor de semblança. Així, els registres podrien ser representats per nodes i les similituds per arestes i es permetria a l'usuari visualitzar i revisar els resultats d'una execució mitjançant un graf, de manera que podria tenir una visió més global d'ells.

- Nova edició de filtres. Tot i que la generació de filtres ha suposat un pas endavant respecte a anteriors versions, seria interessant tenir una creació de filtres mitjançant regles que el mateix usuari incorporaria. Així, s'alliberaria la necessitat de tenir coneixement de *JavaScript* als usuaris experts. A més, seria necessari també que es puguessin editar els diccionaris que poden fer servir els filtres, ja que de moment només n'existeix un i no és modificable.

- Possibilitat d'afegir nous tipus fitxers. Els fitxers d'entrada permesos actualment només són els *CSV* i seria important poder permetre altres tipus, com podria ser la inclusió d'una taula d'una base de dades configurant prèviament els paràmetres de connexió a ella.

- Ampliació ajuda. L'ajuda existent només conté un resum de les funcionalitats més important i seria necessari ampliar-la per a mostrar més detalls.

- Multifiltres. Aquesta opció permetria aplicar filtres a múltiples columnes de manera que es poguessin tractar com una sola columna de creuament (per exemple ajuntar les columnes de dia, mes i any en una de sola que fós data) o també a la inversa, que una columna de dades es fragmentés en múltiples columnes de creuament (per exemple per dividir una columna amb nom complet en les columnes nom, primer cognom i segon cognom).

- Visió de l'auditoria. Tot i que l'auditoria està funcionant perfectament per darrera de l'aplicació, de manera que emmagatzema tots els canvis realitzats per l'usuari, seria interessant que existís una opció per a visualitzar aquesta informació, possiblement només accessible per un administrador.

Apèndix A

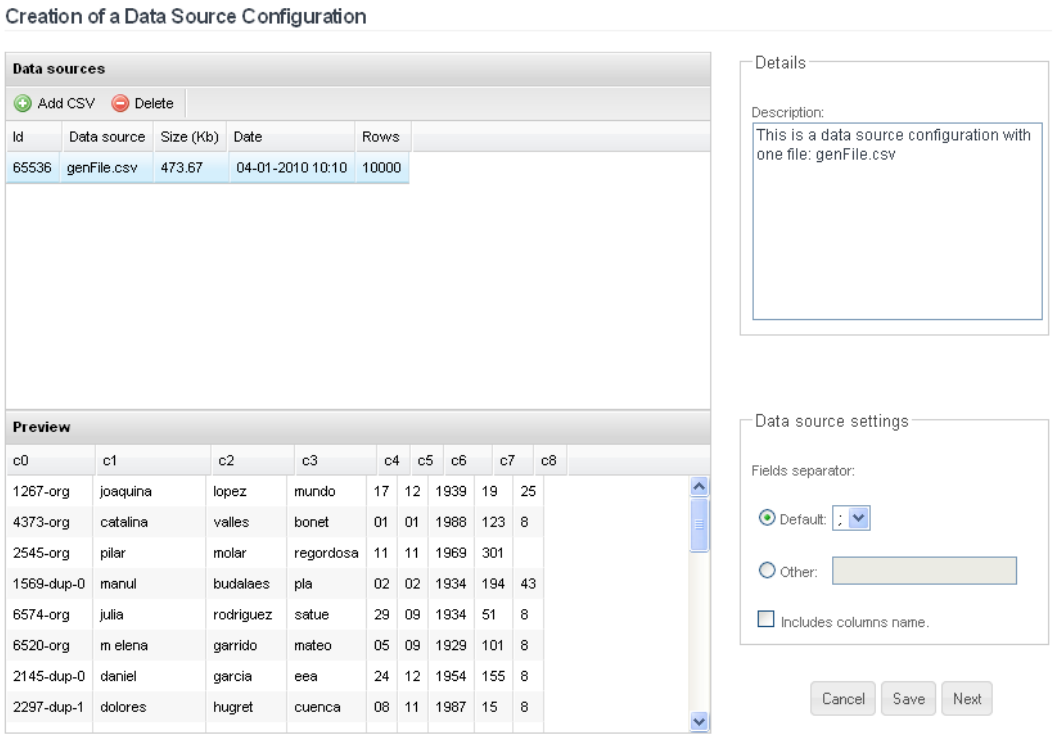
Manual d'ajuda

Record Linkage configuration

Data Source configuration

A Data Source Configuration is used for uploading files to the application. Currently, the accepted format is CSV. In this configuration you will have the possibility to preview the files content and choose a separator character for them. Once a Data Source Configuration is saved, the registers of all its files will be inserted to the database.

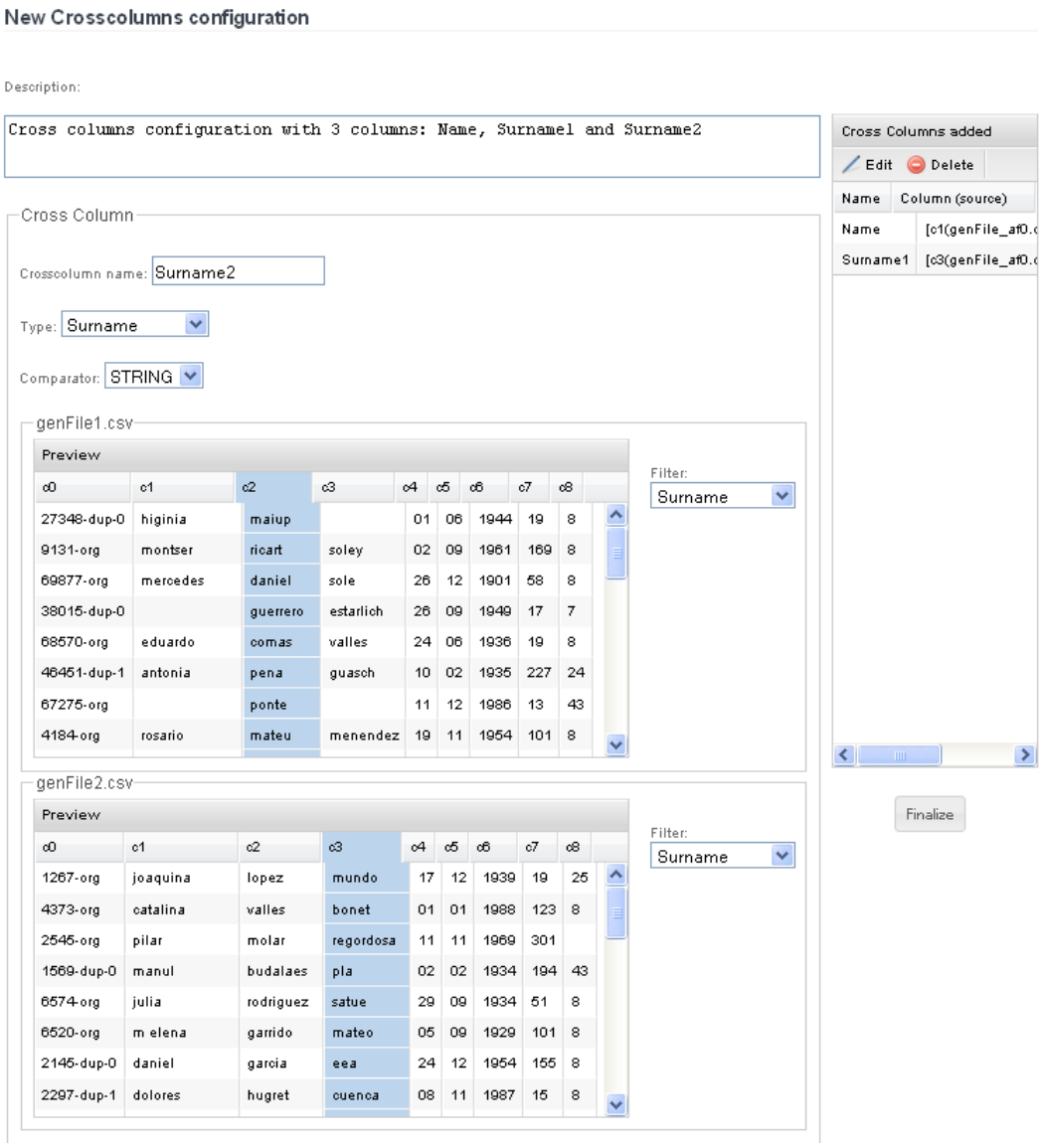
The picture below shows the creation process of this configuration. In the upper table it's possible to see the uploaded sources: a CSV file called genFile.csv and at the bottom it's shown its preview. On the right hand you can see that the separator chosen for this source, which is ';'.



Cross Columns Configuration

A Crosscolumns Configuration is used for choosing the columns that you would like to be processed in the Record Linkage method. Not all the columns of the sources (inserted in a Data Source configuration) are relevant for this process and that's why you can select them in this configuration. Every Crosscolumn of this configuration must be mapped with a column of every source of the configuration and it's also necessary to select a filter for them. Filters help to standardize the values of the registers by cleaning characters which are not useful like for example accents or commas.

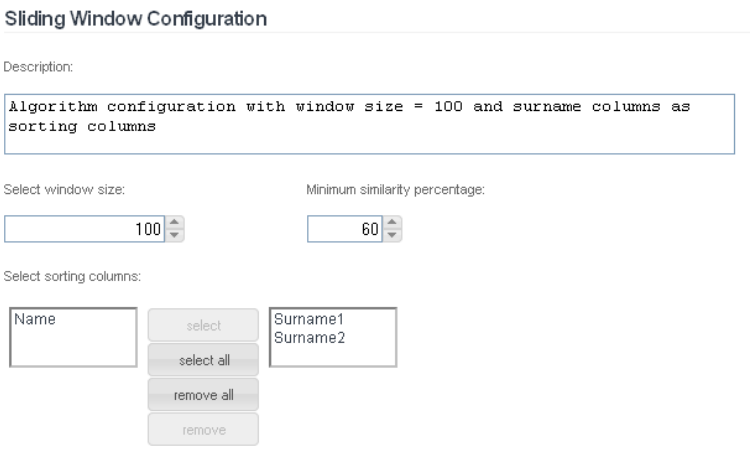
The picture below shows the creation process of this configuration. It's possible to see on the left hand that two crosscolumns have been already created: Name and Surname1. A new crosscolumn called Surname2 is being created and it's mapped with the third column of the source called genFile1.csv and the fourth one of the source called genFile2.csv. Both source columns will be standardized by a filter called Surname.



Algorithm Configuration

A Algorithm Configuration is used for selecting the values of the comparing method. Currently, there's just one method available, which is called Sliding Window. This method is characterized by two attributes: window size and sorting columns. The window size is the number of registers which will be compared with everyone of them, so a register won't be compared with all the others but just to "window size" of them. The sorting columns are a subset of the Crosscolumns selected in the Crosscolumns Configuration. They are used for ordering the registers alphabetically and making comparisons.

The next picture shows the creation process of this configuration. The Crosscolumns Surname1 and Surname2 have been selected as sorting columns and the window size is 100. It's also mandatory to select the minimum similarity percentage that must have the similarities found with this configuration.

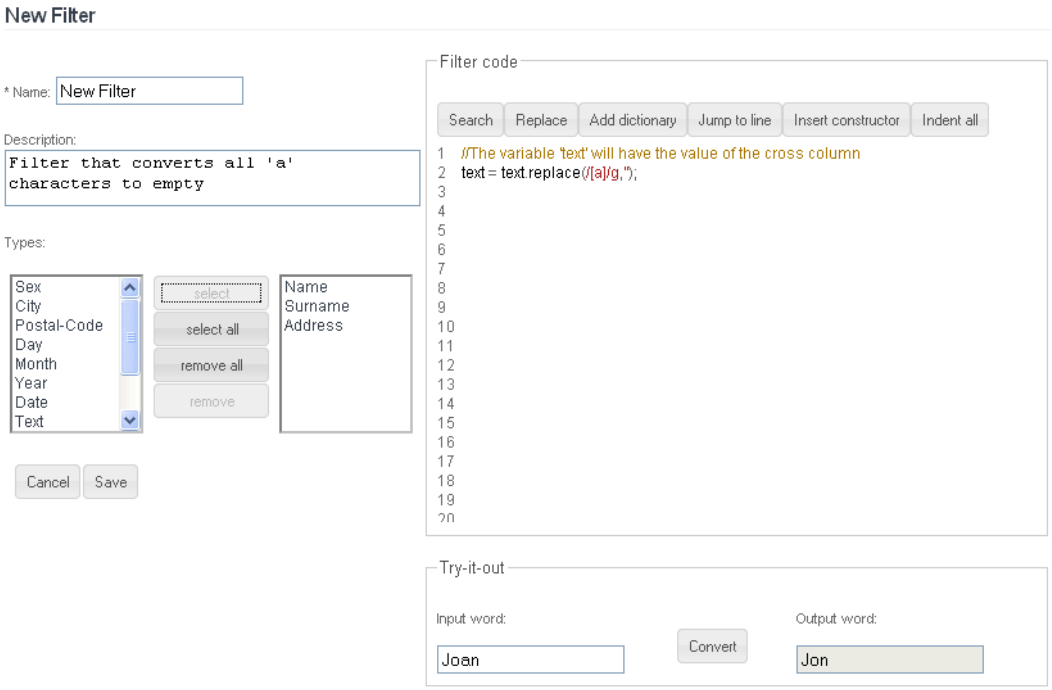


Filters Management

Create and edit filters

Filters are useful for standardizing data. During a Crosscolumn configuration, it's asked what filter will be applied to every column conforming a Crosscolumn. In this system you can add new filters and edit old ones by entering to "Configurations->Filters".

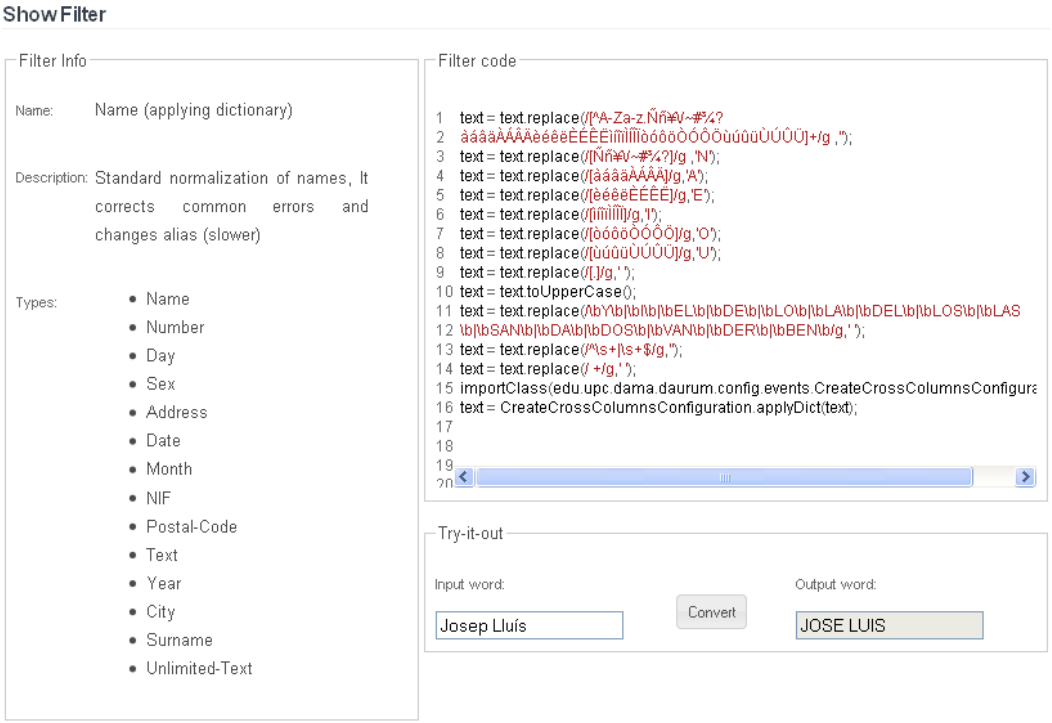
The picture below shows the creation of a new filter, which will suppress all 'a' characters from data. You can see also an example of its execution on the Try-it-out section.



Show filters

You can also see the result of applying an existing filter to a selected word by clicking the "Show" button when watching the filters list.

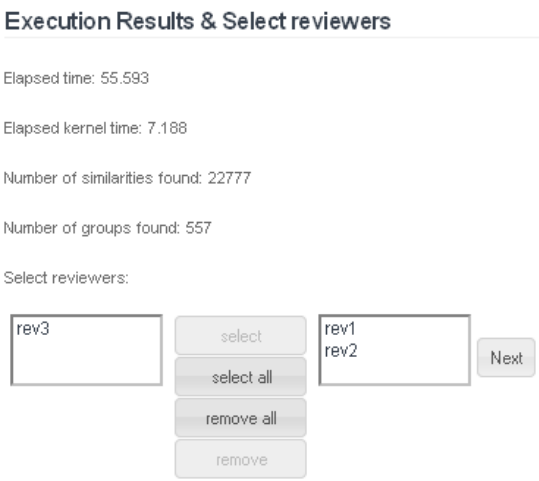
The next picture shows an existing filter and also its execution.



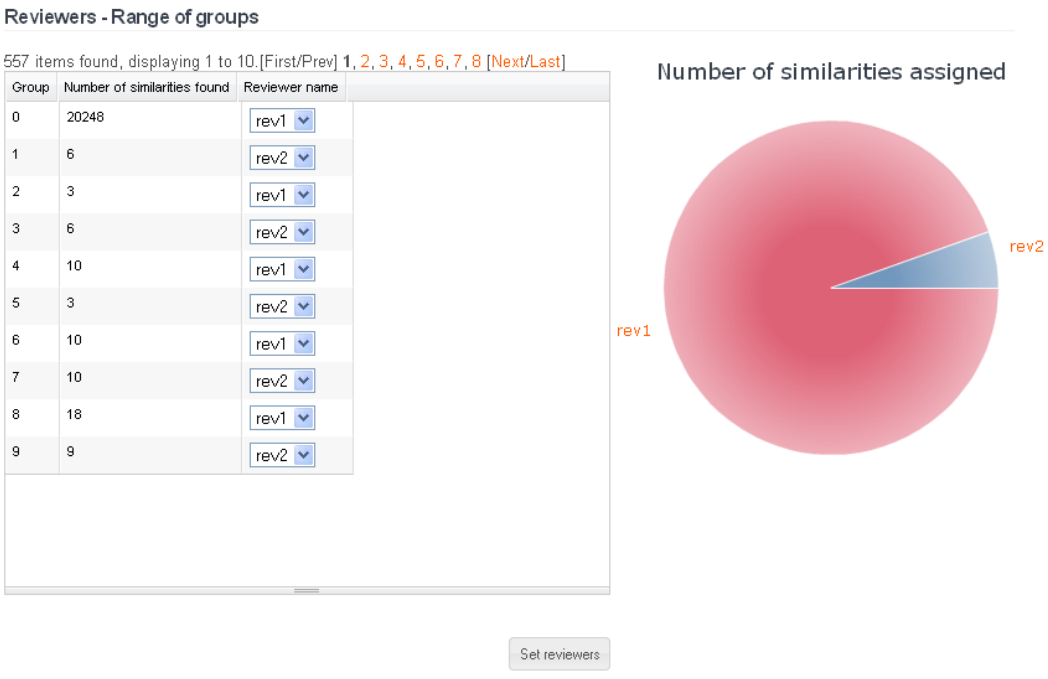
Review Process

Assign reviewers

After executing an algorithm configuration, you can see some information about it. It is also asked the reviewers you would like to assign for the review process. In the next picture you can see an example about it, where two reviewers are going to be assign to an execution which has found 22777 similarities.



After that, the system will automatically assign the groups of similarities to the reviewers selected and it will give the oportunity to change manually these assignments. On the right hand it's shown an statistic about the total percentage of similarities assigned to every reviewer. The picture below shows this process.



Grid Review

After selecting the configuration you would like to review, a grid with similarities is shown. Here, it's possible to change the state of a similarity in three different ways.

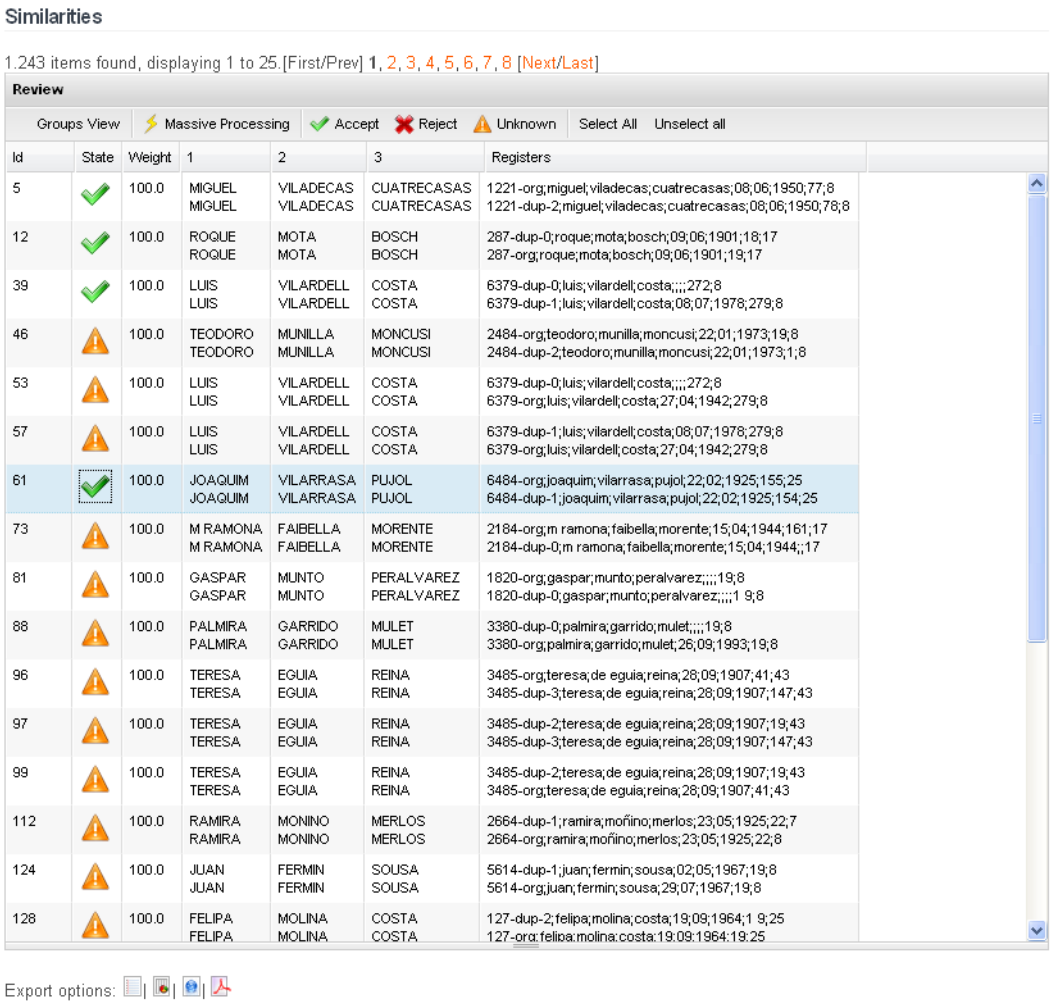
The first one is by clicking on the state image of a similarity. This event will change it to one of the three different states existing in the system: accepted, rejected or unknown.

The second one is by selecting one or more similarities and then clicking on one of the state buttons on the top of the table. You can also select all the similarities shown on the table and change their state by clicking on the "Select All" button.

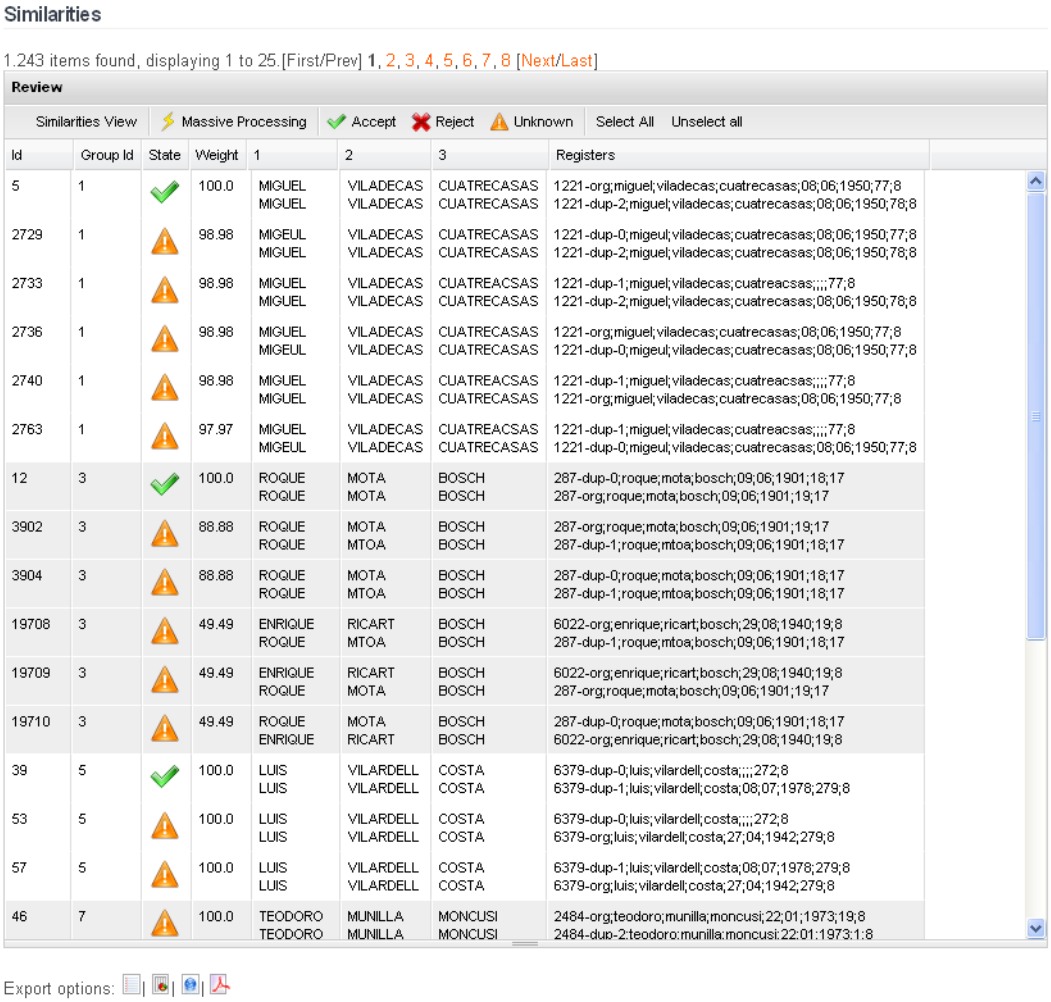
The third one is by doing a massive processing. This way you can select a range of similarity weights and change all of them to the desired state.

There are also two ways of watching the similarities:

The first way is called Similarities View and the similarities found are shown in descending order by their weight. In the next picture you can see an example of it.



The second way is called Groups View and the similarities found are shown in groups, which have different colors and represent the relation between similarities. In the next picture you can see an example of it.



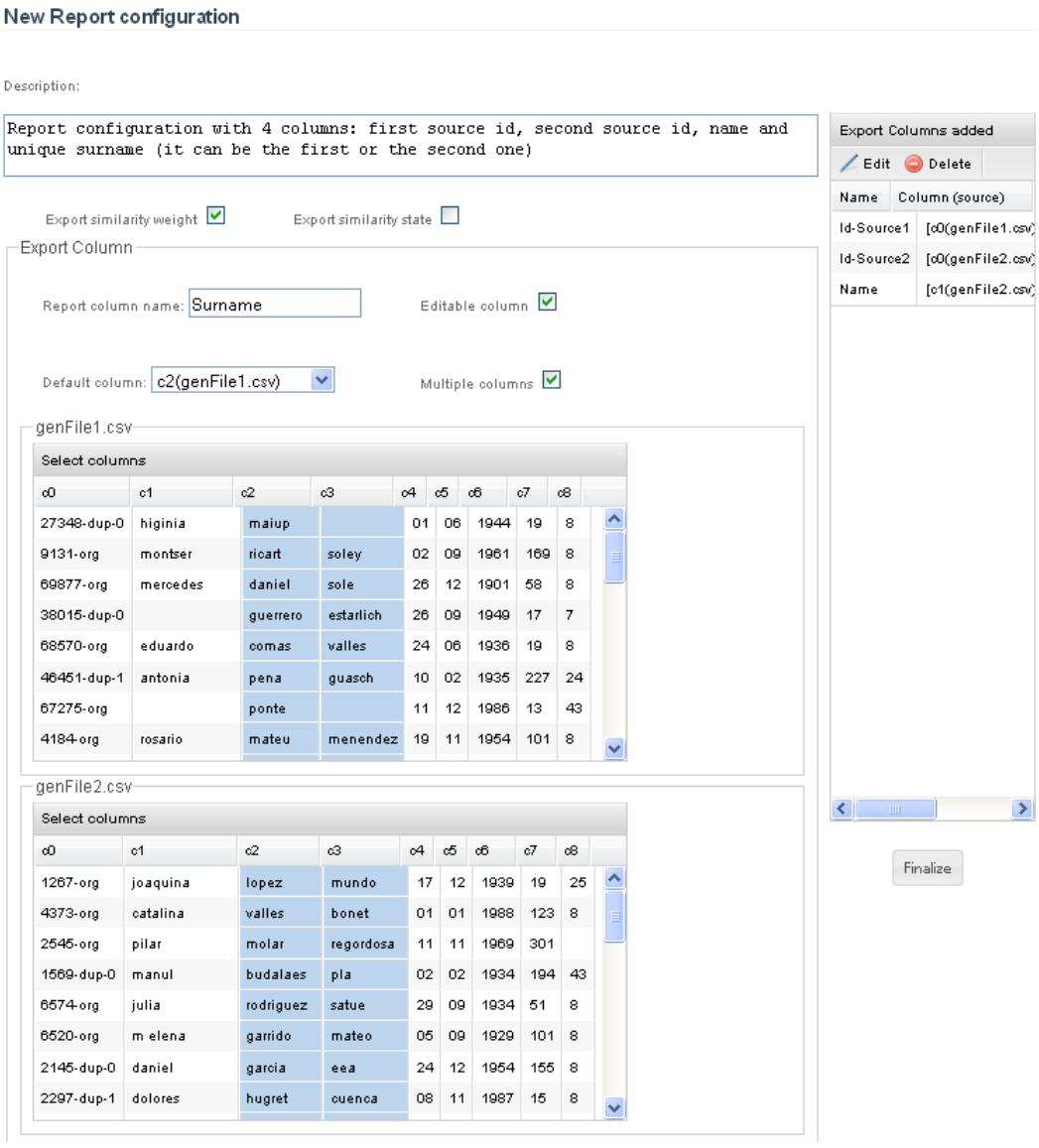
Id	Group Id	State	Weight	1	2	3	Registers
5	1	✓	100.0	MIGUEL MIGUEL	VILADECAS VILADECAS	CUATRECASAS CUATRECASAS	1221-org;miguel;viladecas;cuatrecasas;08;06;1950;77;8 1221-dup-2;miguel;viladecas;cuatrecasas;08;06;1950;78;8
2729	1	⚠	98.98	MIGEUL MIGUEL	VILADECAS VILADECAS	CUATRECASAS CUATRECASAS	1221-dup-0;migeul;viladecas;cuatrecasas;08;06;1950;77;8 1221-dup-2;miguel;viladecas;cuatrecasas;08;06;1950;78;8
2733	1	⚠	98.98	MIGUEL MIGUEL	VILADECAS VILADECAS	CUATREACSAS CUATRECASAS	1221-dup-1;miguel;viladecas;cuatreacsas;;;;77;8 1221-dup-2;miguel;viladecas;cuatrecasas;08;06;1950;78;8
2736	1	⚠	98.98	MIGUEL MIGEUL	VILADECAS VILADECAS	CUATRECASAS CUATRECASAS	1221-org;miguel;viladecas;cuatrecasas;08;06;1950;77;8 1221-dup-0;migeul;viladecas;cuatrecasas;08;06;1950;77;8
2740	1	⚠	98.98	MIGUEL MIGUEL	VILADECAS VILADECAS	CUATREACSAS CUATRECASAS	1221-dup-1;miguel;viladecas;cuatreacsas;;;;77;8 1221-org;miguel;viladecas;cuatrecasas;08;06;1950;77;8
2763	1	⚠	97.97	MIGUEL MIGEUL	VILADECAS VILADECAS	CUATREACSAS CUATRECASAS	1221-dup-1;miguel;viladecas;cuatreacsas;;;;77;8 1221-dup-0;migeul;viladecas;cuatrecasas;08;06;1950;77;8
12	3	✓	100.0	ROQUE ROQUE	MOTA MOTA	BOSCH BOSCH	287-dup-0;roque;mota;bosch;09;06;1901;18;17 287-org;roque;mota;bosch;09;06;1901;19;17
3902	3	⚠	88.88	ROQUE ROQUE	MOTA MTOA	BOSCH BOSCH	287-org;roque;mota;bosch;09;06;1901;19;17 287-dup-1;roque;mtoa;bosch;09;06;1901;18;17
3904	3	⚠	88.88	ROQUE ROQUE	MOTA MTOA	BOSCH BOSCH	287-dup-0;roque;mota;bosch;09;06;1901;18;17 287-dup-1;roque;mtoa;bosch;09;06;1901;18;17
19708	3	⚠	49.49	ENRIQUE ROQUE	RICART MTOA	BOSCH BOSCH	6022-org;enrique;ricart;bosch;29;08;1940;19;8 287-dup-1;roque;mtoa;bosch;09;06;1901;18;17
19709	3	⚠	49.49	ENRIQUE ROQUE	RICART MOTA	BOSCH BOSCH	6022-org;enrique;ricart;bosch;29;08;1940;19;8 287-org;roque;mota;bosch;09;06;1901;19;17
19710	3	⚠	49.49	ROQUE ENRIQUE	MOTA RICART	BOSCH BOSCH	287-dup-0;roque;mota;bosch;09;06;1901;18;17 6022-org;enrique;ricart;bosch;29;08;1940;19;8
39	5	✓	100.0	LUIS LUIS	VILARDELL VILARDELL	COSTA COSTA	6379-dup-0;luis;vilardell;costa;;;;272;8 6379-dup-1;luis;vilardell;costa;08;07;1978;279;8
53	5	⚠	100.0	LUIS LUIS	VILARDELL VILARDELL	COSTA COSTA	6379-dup-0;luis;vilardell;costa;;;;272;8 6379-org;luis;vilardell;costa;27;04;1942;279;8
57	5	⚠	100.0	LUIS LUIS	VILARDELL VILARDELL	COSTA COSTA	6379-dup-1;luis;vilardell;costa;08;07;1978;279;8 6379-org;luis;vilardell;costa;27;04;1942;279;8
46	7	⚠	100.0	TEODORO TEODORO	MUNILLA MONCUSI	MONCUSI MONCUSI	2484-org;teodoro;munilla;moncusi;22;01;1973;19;8 2484-dup-2;teodoro;munilla;moncusi;22;01;1973;1;8

Report

Report configuration

Report configurations are useful for selecting the desired values to export when generating an output file. These configurations give you the chance to choose what values are going to be shown in which output column. You will be able to give a mapping between report columns and the columns of the sources composing a selected Data Source configuration. You will have different ways to map a report column: just one column of every source, multiple columns and also editable columns.

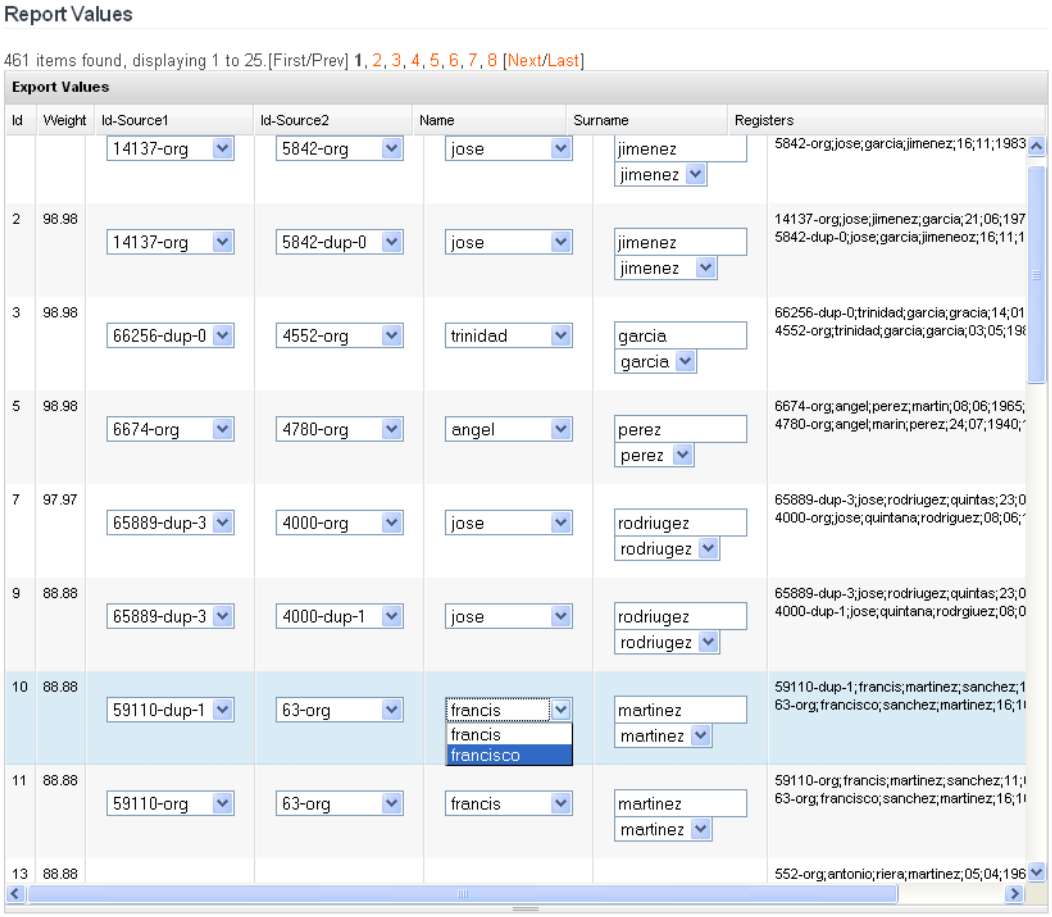
The picture below shows an example of report configurations creation. It has already three columns added and a fourth column is being configured.



Report values

The Report Grid is a way of selecting what values is the user going to export. After selecting the report configuration desired, this grid gives you the chance to select manually what values of every similarity you would like to export. The possible values of every column and similarity will be determinated by the report configuration selected.

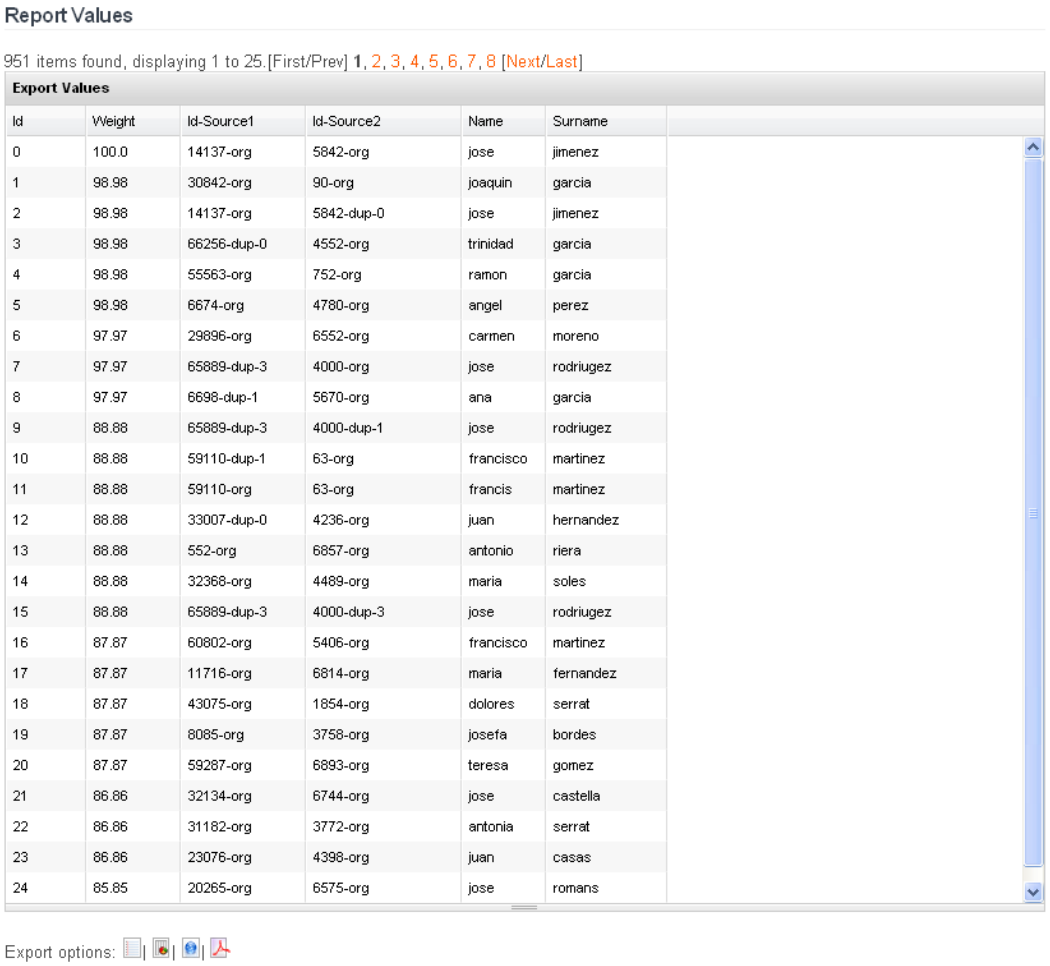
The picture below shows an example of this grid on how to changed a value with a simple box where all possible values are displayed. The similarity with identifier 10 is beeing changed.



Generating report files

After selecting desired the values to export for every similarity, it's time to export them to a physical source. This option gives the chance to export these values to four different formats: CSV, XLS, XML and PDF. A table with all values will be shown and at the bottom there are displayed these different formats. You can export in one of those by clicking on the buttons.

The picture below shows an example of report values and the four different available formats as export options.



Report Values

951 items found, displaying 1 to 25.[First/Prev] 1, 2, 3, 4, 5, 6, 7, 8 [Next/Last]

Export Values

Id	Weight	Id-Source1	Id-Source2	Name	Surname
0	100.0	14137-org	5842-org	jose	jimenez
1	98.98	30842-org	90-org	joaquin	garcia
2	98.98	14137-org	5842-dup-0	jose	jimenez
3	98.98	66256-dup-0	4552-org	trinidad	garcia
4	98.98	55563-org	752-org	ramon	garcia
5	98.98	6674-org	4780-org	angel	perez
6	97.97	29896-org	6552-org	carmen	moreno
7	97.97	65889-dup-3	4000-org	jose	rodriugez
8	97.97	6698-dup-1	5670-org	ana	garcia
9	88.88	65889-dup-3	4000-dup-1	jose	rodriugez
10	88.88	59110-dup-1	63-org	francisco	martinez
11	88.88	59110-org	63-org	francis	martinez
12	88.88	33007-dup-0	4236-org	juan	hernandez
13	88.88	552-org	6857-org	antonio	riera
14	88.88	32368-org	4489-org	maria	soles
15	88.88	65889-dup-3	4000-dup-3	jose	rodriugez
16	87.87	60802-org	5406-org	francisco	martinez
17	87.87	11716-org	6814-org	maria	fernandez
18	87.87	43075-org	1854-org	dolores	serrat
19	87.87	8085-org	3758-org	josefa	bordes
20	87.87	59287-org	6893-org	teresa	gomez
21	86.86	32134-org	6744-org	jose	castella
22	86.86	31182-org	3772-org	antonia	serrat
23	86.86	23076-org	4398-org	juan	casas
24	85.85	20265-org	6575-org	jose	romans

Export options:

Here, it's shown an example of the four available formats with the similarity number 10 highlighted (You can zoom by clicking on them).

- CSV Format

```
0,100.0,14137-org,5842-org,jose,jimenez
1,98.98,30842-org,90-org,joaquin,garcia
2,98.98,14137-org,5842-dup-0,jose,jimenez
3,98.98,66256-dup-0,4552-org,trinidad,garcia
4,98.98,55563-org,752-org,ramon,garcia
5,98.98,6674-org,4780-org,angel,perez
6,97.97,29896-org,6552-org,carmen,moreno
7,97.97,65889-dup-3,4000-org,jose,rodriugez
8,97.97,6698-dup-1,5670-org,ana,garcia
9,88.88,65889-dup-3,4000-dup-1,jose,rodriugez
10,88.88,59110-dup-1,63-org,francisco,martinez
11,88.88,59110-org,63-org,francis,martinez
12,88.88,33007-dup-0,4236-org,juan,hernandez
13,88.88,552-org,6857-org,antonio,riera
14,88.88,32368-org,4489-org,maria,soles
15,88.88,65889-dup-3,4000-dup-3,jose,rodriugez
```

- XLS Format

Id	Weight	Id-Source1	Id-Source2	Name	Surname
0	100.0	14137-org	5842-org	jose	jimenez
1	98.98	30842-org	90-org	joaquin	garcia
2	98.98	14137-org	5842-dup-0	jose	jimenez
3	98.98	66256-dup-0	4552-org	trinidad	garcia
4	98.98	55563-org	752-org	ramon	garcia
5	98.98	6674-org	4780-org	angel	perez
6	97.97	29896-org	6552-org	carmen	moreno
7	97.97	65889-dup-3	4000-org	jose	rodriugez
8	97.97	6698-dup-1	5670-org	ana	garcia
9	88.88	65889-dup-3	4000-dup-1	jose	rodriugez
10	88.88	59110-dup-1	63-org	francisco	martinez
11	88.88	59110-org	63-org	francis	martinez
12	88.88	33007-dup-0	4236-org	juan	hernandez
13	88.88	552-org	6857-org	antonio	riera
14	88.88	32368-org	4489-org	maria	soles
15	88.88	65889-dup-3	4000-dup-3	jose	rodriugez
16	87.87	60802-org	5406-org	francisco	martinez
17	87.87	11716-org	6814-org	maria	fernandez
18	87.87	43075-org	1854-org	dolores	serrat

- XML Format

```
- <row>
    <column>8</column>
    <column>97.97</column>
    <column>6698-dup-1</column>
    <column>5670-org</column>
    <column>ana</column>
    <column>garcia</column>
  </row>
- <row>
    <column>9</column>
    <column>88.88</column>
    <column>65889-dup-3</column>
    <column>4000-dup-1</column>
    <column>jose</column>
    <column>rodriugez</column>
  </row>
- <row>
    <column>10</column>
    <column>88.88</column>
    <column>59110-dup-1</column>
    <column>63-org</column>
    <column>francisco</column>
    <column>martinez</column>
  </row>
- <row>
    <column>11</column>
    <column>88.88</column>
    <column>59110-org</column>
    <column>63-org</column>
    <column>francis</column>
    <column>martinez</column>
  </row>
```

- PDF Format

Id	Weight	Id-Source1	Id-Source2	Name	Surname
0	100.0	14137-org	5842-org	jose	jimenez
1	98.98	30842-org	90-org	joaquin	garcia
2	98.98	14137-org	5842-dup-0	jose	jimenez
3	98.98	66256-dup-0	4552-org	trinidad	garcia
4	98.98	55563-org	752-org	ramon	garcia
5	98.98	6674-org	4780-org	angel	perez
6	97.97	29896-org	6552-org	carmen	moreno
7	97.97	65889-dup-3	4000-org	jose	rodriugez
8	97.97	6698-dup-1	5670-org	ana	garcia
9	88.88	65889-dup-3	4000-dup-1	jose	rodriugez
10	88.88	59110-dup-1	63-org	francisco	martinez
11	88.88	59110-org	63-org	francis	martinez
12	88.88	33007-dup-0	4236-org	juan	hernandez
13	88.88	552-org	6857-org	antonio	riera
14	88.88	32368-org	4489-org	maria	soles
15	88.88	65889-dup-3	4000-dup-3	jose	rodriugez
16	87.87	60802-org	5406-org	francisco	martinez
17	87.87	11716-org	6814-org	maria	fernandez
18	87.87	43075-org	1854-org	dolores	serrat

Apèndix B

Mapeig entitat-relació de la base de dades

```
    <class name="DataSource" table="DATASOURCE">
        <id name="identifier" column="DS_ID" type="integer">
            <generator class="native"/>
        </id>
        <discriminator column="TYPE" type="string"/>
        <property name="name" column="DS_NAME" type="string" />
        <property name="fieldSeparator" column="FIELD_SEPARATOR" type="string" />
        <set name="dataSourceColumns" lazy="true" table="DS_COLUMNS" inverse="true" fetch="join" order-by="INDEX"
cascade="delete">
            <key column="DATA_SOURCE" />
            <one-to-many class="DataSourceColumn" />
        </set>
        <set name="registers" table="REGISTER" lazy="true" inverse="true" fetch="join" order-by="REG_NUM" cascade="delete">
            <key column="DS_ID" />
            <one-to-many class="Register" />
        </set>
        <subclass name="CSVDataSource" discriminator-value="CSV">
            <property name="content" type="blob" lazy="true">
                <column name="CONTENT" sql-type="blob" length="104857600"></column>
            </property>
        </subclass>
    </class>
    <class name="DataSourceColumn" table="DS_COLUMNS" >
        <composite-id >
            <key-many-to-one name="dataSource" column="DATA_SOURCE" class="DataSource" />
            <key-property name="name" column="DSC_NAME" type="string" />
        </composite-id>
        <property name="index" column="INDEX" type="integer"/>
    </class>
    <class name="UploadFile" table="FILES_UPLOADED" >
        <id name="id" column="FILE_ID" type="long">
            <generator class="native"/>
        </id>
        <property name="sessionId" column="HTTP_Session_ID" type="string" />
        <property name="name" column="NAME" type="string" />
        <property name="contentType" type="string" />
        <property name="content" type="blob" >
            <column name="CONTENT" sql-type="blob" length="104857600"></column>
        </property>
    </class>
```

```
    <class name="CrossColumnsConfiguration" table="CROSS_COLUMNS_CONFIGURATION" >
        <id name="identifier" column="CC_CONFIG_ID" type="integer">
            <generator class="native"/>
        </id>
        <property name="description" column="CC_CONFIG_DESCRIPTION" type="string" />
        <property name="createdBy" column="CC_CONFIG_CREATEDBY" type="string" />
        <property name="creationDate" column="CREATION_DATE" type="java.util.Date" update="false" not-null="true" />
        <many-to-one column="DSCONFIG_ID" name="dataSourceConfiguration" class="DataSourceConfiguration" lazy="false"
/>
        <list name="crossColumns" lazy="false" table="CROSS_COLUMNS" inverse="true" fetch="join" cascade="delete">
            <key column="CC_CONFIG" />
            <list-index column="INDEX"></list-index>
            <one-to-many class="CrossColumn"/>
        </list>
    </class>
    <class name="CrossColumn" table="CROSS_COLUMNS">
        <composite-id>
            <key-property name="name" column="CC_NAME" type="string" />
            <key-many-to-one name="crossColumnsConfiguration" class="CrossColumnsConfiguration" >
                <column name="CC_CONFIG" />
            </key-many-to-one>
        </composite-id>
        <set name="filterAssignments" lazy="false" order-by="DATA_SOURCE" inverse="true" fetch="join" cascade="delete"
>
            <key>
                <column name="CC_NAME"></column>
                <column name="CC_CONFIG"></column>
            </key> <one-to-many class="FilterAssignment" />
        </set>
        <property name="weight" column="WEIGHT" type="double" />
        <property name="nullWeight" column="NULL_WEIGHT" type="double" />
        <property name="index" column="INDEX" type="integer"/>
        <many-to-one name="type" column="TYPE" class="CrossColumnType" ></many-to-one>
        <many-to-one name="comparisonType" column="COMPARISON_TYPE" class="ComparisonType" />
    </class>
    <class name="CrossColumnType" table="CROSS_COLUMN_TYPES">
        <id name="name" column="NAME" type="string" />
        <property name="dataType" column="DATA_TYPE" type="integer"></property>
        <property name="matchedProbability" column="MATCHED_PROBABILITY" type="double" />
        <property name="unmatchedProbability" column="UNMATCHED_PROBABILITY" type="double" />
        <property name="needsCache" column="NEEDS_CACHE" type="boolean" />
        <property name="isPredefined" column="IS_PREDEFINED" type="boolean" />
        <many-to-one name="defaultFilter" column="DEFAULT_FILTER" class="Filter" />
        <set name="availableFilters" table="FILTER_APPLICATION_CROSS_COLUMN" lazy="false" >
            <key column="NAME" />
            <many-to-many class="Filter" column="F_NAME"/>
        </set> <many-to-one name="defaultComparisonType" column="DEFAULT_COMPARISON_TYPE" class="ComparisonType"
/>
    </class>
    <class name="Filter" table="FILTERS" mutable="false">
        <id name="name" column="F_NAME" type="string" />
        <property name="code" column="CODE" type="string" length="1000"/>
        <property name="description" column="DESCRIPTION" type="string"/>
        <set name="types" table="FILTER_APPLICATION_CROSS_COLUMN" inverse="true" fetch="join" >
            <key column="F_NAME" />
            <many-to-many class="CrossColumnType" column="NAME"/>
```

```
        </set>
    </class>
    <class name="FilterAssignment" table="FILTER_ASSIGNMENTS">
        <composite-id>
            <key-many-to-one name="crossColumn" class="CrossColumn" >
                <column name="CC_NAME" />
                <column name="CC_CONFIG" />
            </key-many-to-one>
            <key-many-to-one name="dataSourceColumn" class="DataSourceColumn" >
                <column name="DATA_SOURCE" />
                <column name="DSC_NAME" />
            </key-many-to-one>
        </composite-id>
        <many-to-one name="filter" column="FILTER" class="Filter" />
    </class>
    <class name="ComparisonType" table="COMPARISON_TYPE" mutable="false" >
        <id name="name" column="NAME" type="string" />
        <property name="num" column="NUM" type="integer"></property>
    </class>
    <class name="CrossColumnField" table="CROSS_COLUMN_FIELDS">
        <composite-id>
            <key-many-to-one name="register" class="Register" >
                <column name="REG_NUM" />
                <column name="DS_ID" />
            </key-many-to-one>
            <key-many-to-one name="ccConfiguration" class="CrossColumnsConfiguration" >
                <column name="CC_CONFIG_NAME" />
            </key-many-to-one>
        </composite-id>
        <property name="text" column="TEXT" type="string" length="32672"/>
        <property name="row" column="ROW" type="string" length="32672"/>
    </class>
    <class name="Synonym" table="SYNONYM" mutable="false">
        <id name="id" column="ID" type="string"></id>
        <property name="synonym" column="SYN" type="string"></property>
    </class>
    <class name="RecordLinkageConfiguration" table="RL_CONF" >
        <id name="identifier" column="RLCONFIG_ID" type="integer">
            <generator class="native"/>
        </id>
        <discriminator column="ALGORITHM" type="string"/>
        <property name="description" column="RL_CONFIG_DESCRIPTION" type="string" />
        <property name="createdBy" column="RL_CONFIG_CREATEDBY" type="string" />
        <property name="creationDate" column="CREATION_DATE" type="java.util.Date" update="false" not-null="true" />
        <property name="executed" column="EXECUTED" type="boolean" />
        <property name="minimumSimilarityPercentage" column="SIMILARITY_PER" type="double"/>
        <property name="kernelTime" column="KERNEL_TIME" type="double" />
        <property name="totalTime" column="TOTAL_TIME" type="double" />
        <property name="numSimilarities" column="NUM_SIMS" type="int" />
        <property name="numGroups" column="NUM_GROUPS" type="int" />
        <many-to-one column="CC_CONFIG_ID" name="crossColumnsConfiguration" class="CrossColumnsConfiguration" lazy="false"
/>
        <set name="groups" lazy="true" inverse="true" fetch="join" order-by="GROUP_NUM" cascade="delete">
            <key column="RLCONFIG_ID" />
            <one-to-many class="edu.upc.dama.daurum.rlpost.model.Group" />
        </set>
```

```
        <subclass name="SlidingWindowConfiguration" discriminator-value="SLIDING_WINDOW">
                <property name="windowSize" column="WINDOW_SIZE" type="integer" />
                <set name="sortingColumns" table="SORTING_COLUMNS" lazy="false" order-by="CC_NAME">
                        <key column="RLCONFIG_ID"/>
                        <many-to-many class="CrossColumn">
                                <column name="CC_NAME"></column>
                                <column name="CC_CONFIG"></column>
                        </many-to-many>
                </set>
        </subclass>
</class>
<class name="Group" table="GROUP_SIM" >
        <composite-id>
                <key-property name="num" column="GROUP_NUM" type="long" />
                <key-many-to-one name="rlConfiguration" class="RecordLinkageConfiguration" >
                        <column name="RLCONFIG_ID" />
                </key-many-to-one>
        </composite-id>
        <many-to-one name="reviewer" column="REVIEWER" class="User"></many-to-one>
        <set name="similarities" lazy="false" inverse="true" fetch="join" cascade="delete">
                <key>
                        <column name="GROUP_NUM"></column>
                        <column name="RLCONFIG_ID"></column>
                </key> <one-to-many class="Similarity"/>
        </set>
</class>
<class name="Similarity" table="SIMILARITY">
        <composite-id>
                <key-property name="num" column="NUM" type="long" />
                <key-many-to-one name="group" class="Group" >
                        <column name="GROUP_NUM" />
                        <column name="RLCONFIG_ID"/>
                </key-many-to-one>
        </composite-id>
        <property name="weight" column="WEIGHT" type="double" not-null="false" />
        <many-to-one name="firstRegister" class="Register" lazy="false">
                <column name="FIRST_REG"></column>
                <column name="DS_ID1"></column>
        </many-to-one>
        <many-to-one name="secondRegister" class="Register" lazy="false">
                <column name="SECOND_REG"></column>
                <column name="DS_ID2"></column>
        </many-to-one>
        <property name="state" column="STATE" type="string"></property>
</class>
<class name="Register" table="REGISTER">
        <composite-id>
                <key-property name="num" column="REG_NUM" type="long" />
                <key-many-to-one name="dataSource" class="DataSource">
                        <column name="DS_ID" />
                </key-many-to-one>
        </composite-id>
        <property name="text" column="TEXT" type="string" length="32672"/>
        <property name="datasourceId" column="DS_ID" insert="false" update="false"/>
</class>
<joined-subclass name="Reviewer" table="USUARIS" lazy="true" extends="User">
```

```
        <subselect>
            select USUARIS.NAME,USUARIS.PASSWORD, USUARIS.FULL_NAME, USUARIS.EMAIL, USUARIS.TELEPHONE,
USUARIS.DIRECTION, USUARIS.JOINING_DATE from USUARIS, USER_ROLES where USER_ROLES.ROLE_ID='reviewer' and
USUARIS.NAME=USER_ROLES.NAME
        </subselect>
        <synchronize table="USUARIS"/>
        <synchronize table="USER_ROLES"/>
        <key column="NAME" />
    </joined-subclass>
    <class name="ExportConfiguration" table="EXPORT_CONFIGURATION" >
        <id name="identifier" column="ECONFIG_ID" type="integer">
            <generator class="native"/>
        </id>
        <property name="description" column="ECONFIG_DESCRIPTION" type="string" />
        <property name="createdBy" column="ECONFIG_CREATEDBY" type="string" />
        <property name="showState" column="SHOW_STATE" type="boolean" />
        <property name="showWeight" column="SHOW_WEIGHT" type="boolean" />
        <property name="creationDate" column="CREATION_DATE" type="java.util.Date" update="false" not-null="true" />
        <many-to-one column="DSCONFIG_ID" name="dataSourceConfiguration" class="DataSourceConfiguration" lazy="false"
/>
        <list name="exportColumns" table="EXPORT_COLUMNS" lazy="true" inverse="true" fetch="join" cascade="delete">
            <key column="ECONFIG" />
            <list-index column="INDEX"></list-index>
            <one-to-many class="ExportColumn"/>
        </list>
    </class>
    <class name="ExportColumn" table="EXPORT_COLUMNS">
        <composite-id>
            <key-property name="name" column="EC_NAME" type="string" />
            <key-many-to-one name="exportConfiguration" class="ExportConfiguration" >
                <column name="ECONFIG" />
            </key-many-to-one>
        </composite-id>
        <set name="columnsRange" lazy="false" order-by="DATA_SOURCE" >
            <key>
                <column name="EC_NAME"></column>
                <column name="ECONFIG"></column>
            </key>
            <many-to-many class="DataSourceColumn">
                <column name="DATA_SOURCE"></column>
                <column name="COL_NAME"></column>
            </many-to-many>
        </set>
        <property name="index" column="INDEX" type="integer"/>
        <property name="editable" column="EDITABLE" type="boolean" />
        <many-to-one name="defaultColumn" class="DataSourceColumn" lazy="false">
            <column name="DEFAULT_DATA_SOURCE"></column>
            <column name="DEFAUL_COL_NAME"></column>
        </many-to-one>
    </class>
    <class name="ExportColumnField" table="EXPORT_COLUMN_FIELDS">
        <composite-id>
            <key-many-to-one name="similarity" class="edu.upc.dama.daurum.rlpost.model.Similarity" >
                <column name="NUM" />
                <column name="GROUP_NUM" />
                <column name="RLCONFIG_ID" />
```

```
                </key-many-to-one>
                <key-many-to-one name="exportConfiguration" class="ExportConfiguration" >
                        <column name="ECONFIG_ID" />
                </key-many-to-one>
        </composite-id>
        <property name="text" column="TEXT" type="string" length="32672"/>
</class>
<class name="User" table="USUARIS" polymorphism="explicit">
        <id name="name" column="NAME" type="string" length="10"/>
        <property name="password" type="string" access="field">
                <column name="PASSWORD" not-null="true" length="16"></column>
        </property>
        <property name="fullName" type="string" access="field">
                <column name="FULL_NAME" not-null="false" length="50"></column>
        </property>
        <property name="email" type="string" access="field">
                <column name="EMAIL" not-null="false" length="20"></column>
        </property>
        <property name="telephone" type="string" access="field">
                <column name="TELEPHONE" not-null="false" length="20"></column>
        </property>
        <property name="direction" type="string" access="field">
                <column name="DIRECTION" not-null="false" length="50" ></column>
        </property>
        <property name="joiningDate" type="java.util.Date" access="field" >
                <column name="JOINING_DATE" not-null="false"></column>
        </property>
        <set name="role" table="USER_ROLES" lazy="false" >
                <key column="NAME" />
                <many-to-many class="Role" column="ROLE_ID"/>
        </set>
</class>
<class name="Role" table="ROLES" mutable="false">
        <id name="name" column="ROLE" type="string"/>
</class>
<class name="Error" table="ERRORS" mutable="false">
        <id name="id" column="ERROR_ID" type="long">
                <generator class="native"/>
        </id>
        <property name="message" update="false" type="string" length="1000" />
        <property name="detail" update="false" type="string" length="1000"/>
        <property name="time" type="timestamp" update="false" insert="false" generated="insert"/>
</class>
```

Apèndix C

Acrònims

AC Arquitectura de Computadors.

BD Base de Dades.

CGI *Common Gateway Interface*.

CSV *Comma Separated Value*.

DAMA-UPC Data Management Group - Universitat Politècnica de Catalunya.

HTML *HyperText Markup Language*.

HTTP *HyperText Transfer Protocol*.

IDE *Integrated Development Environment*.

JDBC *Java DataBase Connectivity*.

JEE *Java Enterprise Edition*.

JNI *Java Native Interface*.

JSP JavaServer Page.

JVM *Java Virtual Machine*.

MVC Model Vista Controlador.

ORM *Object-Relational Mapping*.

PDF *Portable Document Format*.

RL *Record Linkage*.

SB *Standard Blocking*.

SQL *Structured Query Language*.

SW *Sliding Window*.

UML *Unified Modeling Language*.

URL *Uniform Resource Locator*.

XLS Format *Excel*.

XML *eXtensible Markup Language*.

W3C *World Wide Web Consortium*.

Bibliografia

[1] Javaserver pages technology, http://java.sun.com/products/jsp/.

[2] Jquery: The write less, do more, http://jquery.com/.

[3] Apache maven 2, http://maven.apache.org/.

[4] Apache struts 2, http://struts.apache.org/2.1.6/index.html.

[5] Apache tiles 2, http://tiles.apache.org/.

[6] Apache tomcat, http://tomcat.apache.org/.

[7] Dama-upc, http://www.dama.upc.edu.

[8] Eclipse, http://www.eclipse.org/.

[9] Developer resources for java technology, http://www.java.sun.com.

[10] Lyx: The document processor based on tex/latex, http://www.lyx.org/.

[11] W3c, http://www.w3c.es/.

[12] Bauer, C., and King, G. *Java Persistence with Hibernate. Revised Edition of Hibernate in Action; rev. version*. Manning, 2007.

[13] Baxter, R., and Christen, P. A comparison of fast blocking methods for record linkage. pp. 25–27.

[14] Cohen, F. *Java Testing and Design: From Unit Testing to Automated Web Tests*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[15] Deen, S. M., Amin, R. R., and Taylor, M. C. Data integration in distributed databases. *IEEE Trans. Softw. Eng. 13*, 7 (1987), 860–864.

[16] Do, H.-H., and Rahm, E. Coma: a system for flexible combination of schema matching approaches. In *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases* (2002), VLDB Endowment, pp. 610–621.

[17] Fellegi, I. P., and Sunter, A. B. A theory for record linkage. *Journal of the American Statistical Association 64*, 328 (1969), 1183–1210.

[18] Guisado-Gámez, J., Prat-Pérez, A., Nin, J., Muntés-Mulero, V., and Larriba-Pey, J. L. Parallelizing record linkage for disclosure risk assessment. In *PSD '08: Proceedings of the UNESCO Chair in data privacy international conference on Privacy in Statistical Databases* (Berlin, Heidelberg, 2008), Springer-Verlag, pp. 190–202.

[19] Michalowski, M., Thakkar, S., and Knoblock, C. Exploiting secondary sources for unsupervised record linkage. In *Proceedings of the 2004 VLDB Workshop on Information Integration on the Web* (2004).

[20] Rahm, E., and Bernstein, P. A. A survey of approaches to automatic schema matching. *The VLDB Journal 10*, 4 (2001), 334–350.

[21] Rohan Baxter, L. G. Decision models for record linkage. In *Data Mining* (2006), pp. 146–160.

[22] Roughley, I. *Starting Struts 2*. Lulu.com, 2007.

[23] Winkler, W. Data cleaning methods, 2003.

[24] Yan, S., Lee, D., Kan, M.-Y., and Giles, L. C. Adaptive sorted neighborhood methods for efficient record linkage. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries* (New York, NY, USA, 2007), ACM, pp. 185–194.